

UNCLASSIFIED

AD NUMBER

AD481734

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies and their contractors;
Administrative/Operational Use; MAY 1962. Other requests shall be referred to U.S. Naval Postgraduate School, Monterey, CA.

AUTHORITY

USNPS ltr, 6 Oct 1971

THIS PAGE IS UNCLASSIFIED

NPS ARCHIVE
1962
JAUREGUI, S.

A THEORETICAL STUDY OF
COMPLEMENTARY BINARY CODE SEQUENCES AND
A COMPUTER SEARCH FOR NEW KERNELS

STEPHEN JAUREGUI, JR.

LIBRARY
U.S. NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

A THEORETICAL STUDY OF
COMPLEMENTARY BINARY CODE SEQUENCES
AND A COMPUTER SEARCH FOR NEW KERNELS

by

Stephen Jauregui, Jr.

A THEORETICAL STUDY OF
COMPLEMENTARY BINARY CODE SEQUENCES
AND A COMPUTER SEARCH FOR NEW KERNELS

by

Stephen Jauregui, Jr.
Lieutenant, United States Navy

Submitted in partial fulfillment of
the requirements for the degree of

DOCTOR OF PHILOSOPHY

United States Naval Postgraduate School
Monterey, California

May, 1962

A THEORETICAL STUDY OF
COMPLEMENTARY BINARY CODE SEQUENCES
AND A COMPUTER SEARCH FOR NEW KERNELS

by

Stephen Jauregui, Jr.

This work is accepted as fulfilling
the dissertation requirements for the degree of

DOCTOR OF PHILOSOPHY

from the

United States Naval Postgraduate School

ACKNOWLEDGMENTS

The writer wishes to express his gratitude to the various members of the United States Naval Postgraduate School faculty and staff who assisted him in many ways during the investigation. Among these are Professor C.F. Klammer, Jr. who suggested the group theory approach and Professor W.R. Church who assisted in the identification of the group of operations with a known group.

Thanks are extended to Mr. C.W. Erickson (U.S. Navy Electronics Laboratory) who first brought complementary sequences to my attention and also to Professor G.A. Gray who was helpful in several discussions during the investigation. The writer is also indebted to the Assistant Director of Research, Professor C.E. Menneken, who made large blocks of computer time available and to the computer center staff who saved hours of labor by accomplishing some of the routine tasks. Mrs. Bea S. Green is extended thanks for her efforts in typing this dissertation from a much corrected rough draft.

A special expression of gratitude is reserved for Professor M. J. E. Golay of Eindhoven University. His basic work in the field and personal correspondence served as an inspiration to the writer and his wise counsel aided greatly in the search for new codes.

ABSTRACT

Complementary binary sequences were invented by Golay in an investigation of infra-red multi-slit spectrometry. This dissertation formalizes the basic results obtained by Golay and develops new concepts and techniques for examining the characteristics of these special binary codes. This work has developed new understanding of the structure and methods for the decomposition of complementary sequences.

Complementary sequences have the property of an infinite correlation peak to ambiguity ratio when detected with a matched filter. These binary sequences should find much application as pseudo-random noise modulation signals for both radar and communications systems.

A discussion of the need for such sequences is included in the introduction and is followed by a state of the art description. An operations group on the sequences is formulated and the proofs of several theorems concerning the operations group are given in a rigorous manner. One reason for developing the operations group is the application to elimination of redundancy in the computer search for new codes.

Several invariant properties of complementary codes are proved through the use of the Hamming distance concept. Many more invariant properties of the sequences are demonstrated through the introduction of a Hamming vector. The concept of a Hamming vector is extremely useful as a complementary code decomposition tool. A large number of theorems are proved to enhance its use in this field.

Several computer searches for complementary sequences are described and the actual computer programs for the CDC 1604 are included in the Appendix.

TABLE OF CONTENTS

Chapter		Page
I	Introduction	1
II	Complementary Sequences	13
III	The Group of Operations	31
IV	Hamming Distance of Complementary Sequences	49
V	The Hamming Vectors of Complementary Sequences	62
VI	Supplementary and Cyclic Complementary Codes	85
VII	An Exhaustive Search for Kernels of Length 26	89
VIII	A Partial Search for Kernels of Length 34	100
IX	Conclusions	111
	Glossary of Symbols and Terms	118
	Bibliography	119
	Explanation of Format of the Appendices	121
Appendix I	Matched Filters	122
Appendix II	Codes of Length 16	125
Appendix III	Codes of Length 20	129
Appendix IV	Subgroups of the Operations Group	130
Appendix V	$n=26$ Computer Program	135
Appendix VI	$n=34$ Computer Program	151

LIST OF ILLUSTRATIONS

Figure		Page
1.1	Frequency Modulation of a Chirp Radar	4
1.2	Signal Addition Through the Use of a Delay Line	5
1.2	Matched Filter Detection of a Random Code	7
1.4	Matched Filter Detection of an Optimal Noise Code	7
1.5	Matched Filter Detection of a Complementary Code of Length 8	9
2.1	Matched Filter Detection of a Complementary Pair of Length n	14
4.1	Foldover Parity Check Method	57
4.2	Sequence Quadruple Form of Parity Check	58
7.1	Flow Chart for Computer Search for Kernels of Length 26	93
7.2	Number of Codes within each Supplementary Block for $n=26$	91
7.3	Mask of Digits for Likes, Actual Numbers for $n=26$	99
8.1	Flow Chart for Cyclic Complementary Designation	104

LIST OF TABLES

Table		Page
2.1	All possible complementary code lengths up to 200 with unordered possible weights.	25
2.2	All possible binary numbers of length 4.	26
3.1	Some operations.	32
3.2	Elements of the unordered operations group.	35
3.3	Multiplication Table of Operations.	36
3.4	Redundance due to $(I \ II \ I \ \bar{I})$.	44
3.5	Redundance due to $(I \ I \ I \ \bar{I})$.	46
4.1	Some Hamming distances and vectors of codes of length 8.	55
5.1	Hamming vectors of pairs.	74
5.2	All possible Hamming vectors of kernels of length 10.	79
5.3	All possible Hamming vectors of the kernel of length 2^6 .	82
9.1	Noise uniformity of a complementary pair of length 8.	115

CHAPTER I

INTRODUCTION

Search radars of today are required to obtain smaller and smaller targets at ever increasing ranges. This has led to many different techniques of summing radio frequency energy to obtain these long ranges. Complementary sequences are well adapted for coding the RF energy pulses to give a summing of the returning RF energy in the echo. The present work is more concerned with the characteristics of complementary codes such as their operation groups, Hamming distances and Hamming vectors, than it is with their application as modulating signals for RF carriers. However, the introduction will develop the radar problem as a vehicle to emphasize the use of these codes in future applications in radar. Their application will probably be equally important in the communications field. The initial work in complementary sequences was directed toward an infra-red application, but it is felt that the eventual application of these sequences will spread to many different fields.^{1, 2}

The usual approach to the problem of increasing radar range is to look for ways of varying some of the parameters in the radar range equation which will lead to a greater maximum range.

The radar range equation is:

$$R_{\max} = \left[\frac{P_t G A_e S D}{P_{\min} 16\pi^2} \right]^{1/4}$$

where P_t = peak transmitted power in watts,

P_{\min} = minimum peak detectable signal in watts,

S = scattering cross section of target in units
consistent with range,

G = gain of transmitting antenna,

A_e = effective area of receiving antenna in
consistent units,

D = a composite loss factor for transmission lines, atmospheric losses, etc.

The parameters under the control of the design engineer are A_e , G , P_{\min} and P_t . A_e and G are fixed by various considerations such as antenna beamwidth, side lobe suppression, physical size of the antenna, etc., and are not truly available as adjustable parameters beyond small variations. Therefore the two parameters for the designers to optimize are P_{\min} and P_t .

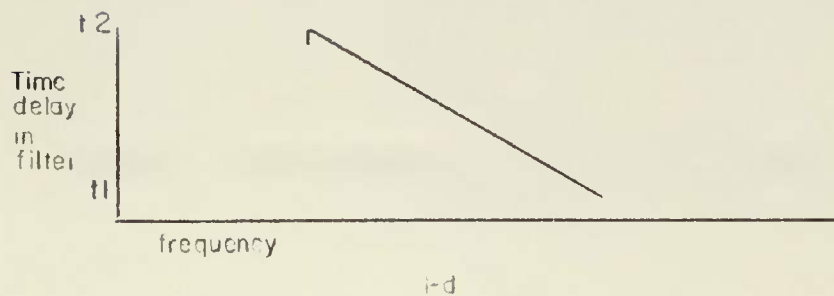
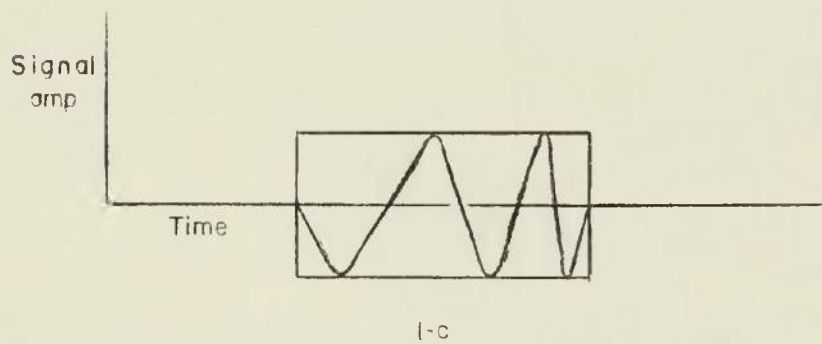
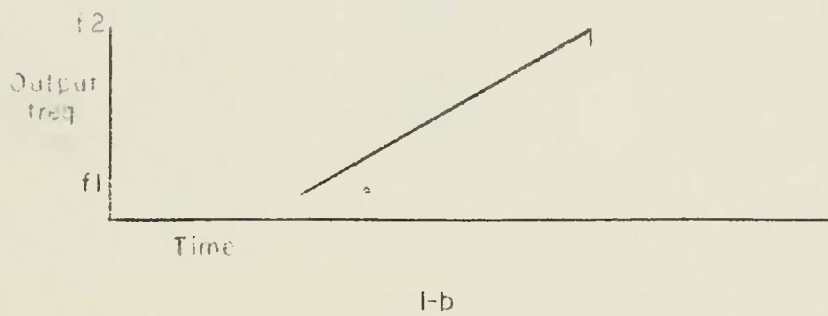
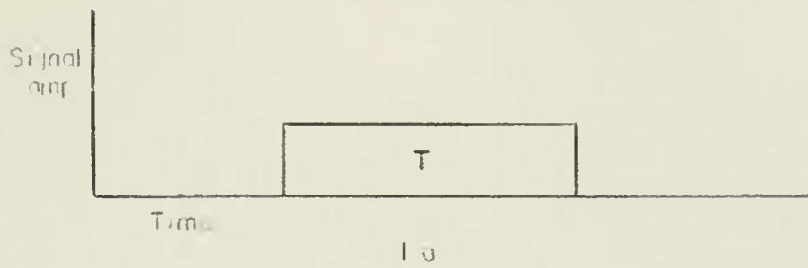
P_{\min} is basically limited by three factors: Johnson noise in the input circuits, shot effect and other noises in the first tube, and cosmic noise picked up by the antenna. P_{\min} approaches KBT_a in a perfect receiver, where K is Boltzmann's constant, B is the bandwidth and T_a is the absolute space temperature. P_{\min} is currently being reduced toward the above value through the utilization of parametric amplifiers, masers and other low noise devices. Another method of reducing P_{\min} is through pulse integration, since noise voltage averaged over a period of time has a value approaching zero while the signal values are additive. However, in most cases this integration of pulses can also be considered as an increase in transmitted energy. It will be considered from that viewpoint in this paper. There are two basic types of integration, coherent and noncoherent. Coherent integration gives a gain of approximately N , where N is the number of pulses integrated and noncoherent integration gives a gain of approximately \sqrt{N} . Both of these values are compared to a single pulse³. The values are respectively the upper bound for coherent integration and the lower bound for noncoherent integration.

There are basically three different integration time bases possible in a radar system. One is the antenna scan to antenna scan where the

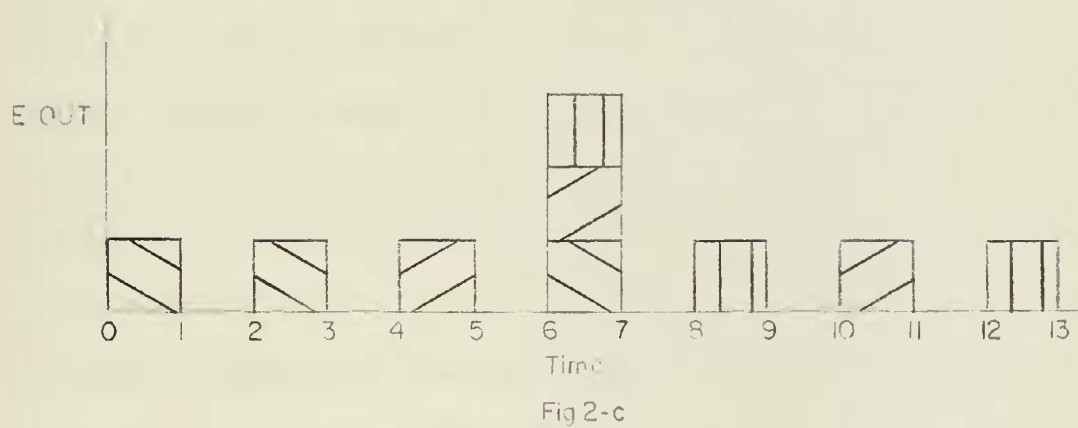
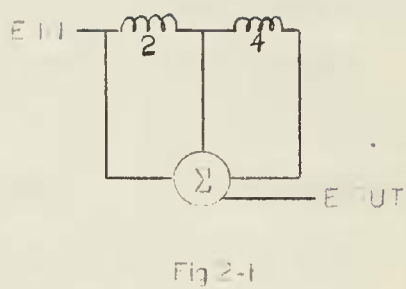
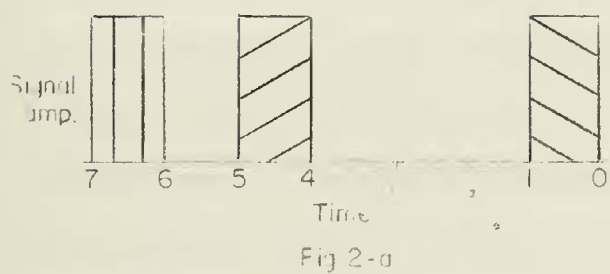
operator or a computer, decides if the echoes from previous scans were signals or noise. The second method of integration is the interpulse method where the cathode ray tube, a delay line, or a computer sums up the hits within one antenna scan of the target. The third method is intrapulse integration, or pulse compression, where different portions of one radar transmitted pulse are summed up. Complementary codes are used in one scheme of intrapulse integration.

An examination of how the transmitted power or energy since $E_t = P_t \Delta T$, can be varied is now in order. From this relation, it is seen that there are two basic methods for increasing the transmitted energy: either increase peak pulse power, or increase the pulse width. A common method of increasing radiated energy has been to increase the peak power. This method has proven both expensive and wasteful as many radar transmitter tubes are peak power limited while only dissipating a very small percentage of their average power capability, and radar modulators are taxed with a very similar problem. The alternative did not look very promising at first, since to increase pulse duration normally reduces the high frequency content of a pulse, which in turn reduces range resolution capabilities. However, intrapulse modulation retains the high frequency content of a narrow pulse, while increasing the energy content to that of a pulse of long duration. Intrapulse modulation allows the summing, required in intrapulse integration, for detection in a radar receiver.

There are two methods of intrapulse modulation, the digital method and the analog method. Although the emphasis in this paper is on digital modulation schemes - the analog system will now be briefly mentioned since it is a part of the overall picture. Figure 1.1 contains the graphs used in the explanation of the chirp radar scheme⁴. In



Frequency modulation of a chirp radar.
FIGURE 1.1



Signal addition through the use of a delay line.

FIGURE 1.2

this analog method of intrapulse modulation the frequency of the outgoing pulse is swept linearly upward with time as is shown in Figure 1.1-b. This gives a large high frequency content to the pulse while at the same time maintaining a long pulse duration. Upon reception the signal is passed through a reversed time delay, delaying the low frequencies more than the highs, as is shown in Figure 1.1-d. This allows all the frequency components of energy to arrive at the output simultaneously, giving a large and narrow output pulse. The type of filter used in the delaying process is known as a matched filter and is discussed in Appendix I. The procedure here is seen to be quite simple: first we generate energy components with various frequencies at different times, and then we delay all components so that they arrive at the output simultaneously.

As a preliminary to the digital type intrapulse modulation consider a transmitted waveform such as in Figure 1.2. This waveform consists of three pulses of unit length with spacing between them of one and three time units respectively. This waveform is then fed into a delay line in the receiver which has the delays shown in Figure 1.2-b. The output is either a one or a zero except at the exact match of signal to filter; the output is then three as is shown in Figure 1.2-c. The three pulses have therefore been summed as though they were one, at the time between 6 and 7 units. These pulses still contain all the high frequency components of the short pulse of unit length but have effectively three times the energy transmitted in a single pulse at the time of exact match. There are two features to be noted about this output, first it has been delayed in time and second there are secondary pulses at various undesired times which might be classified as coding noise.

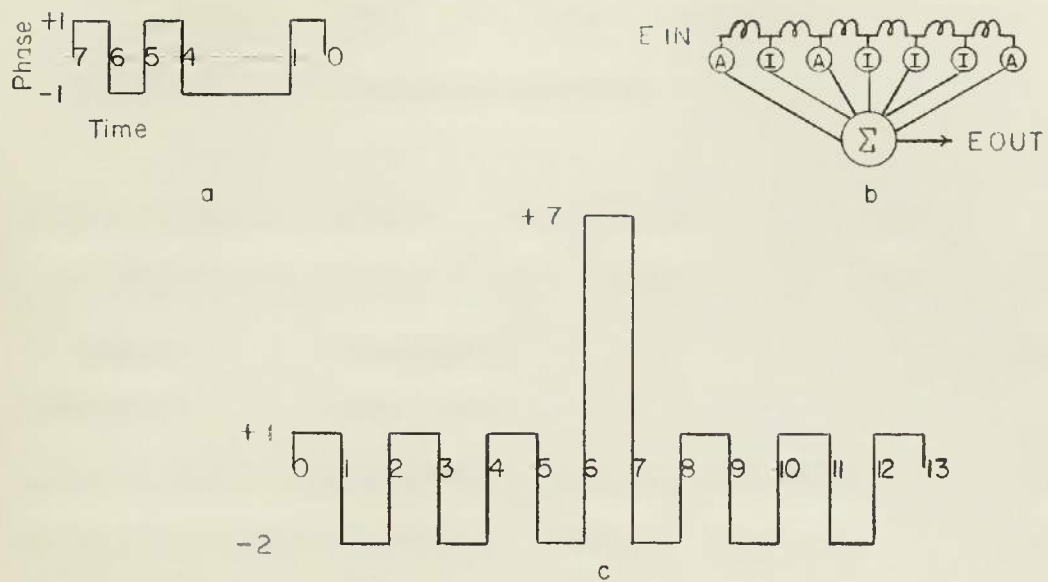


FIGURE 1.3

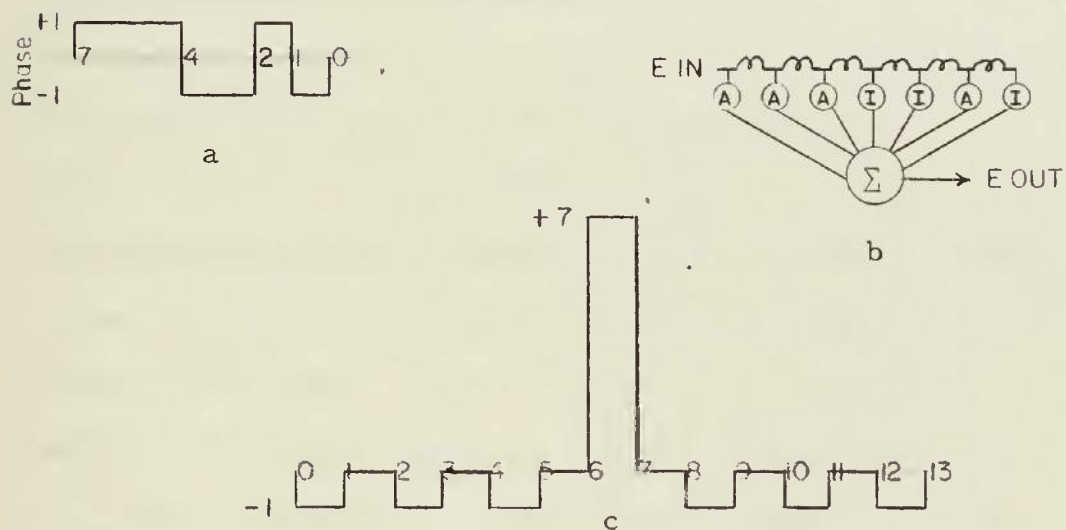


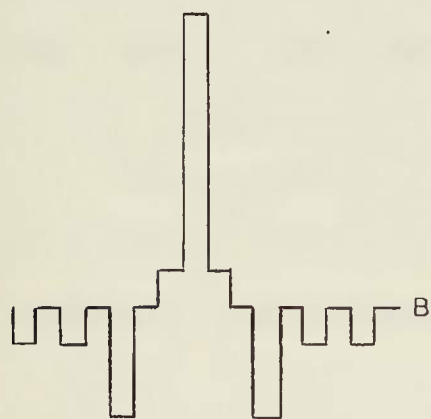
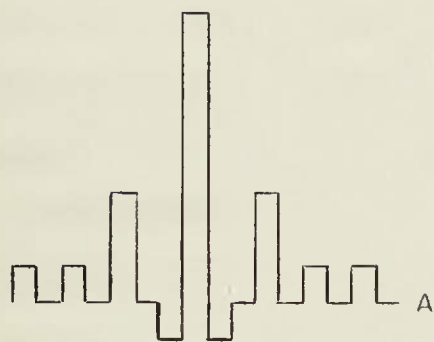
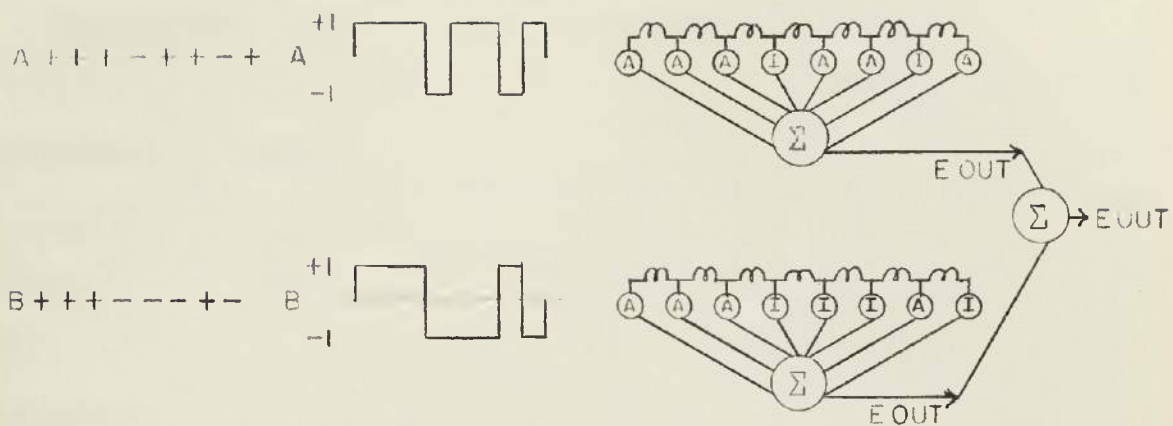
FIGURE 1.4

Fig. 1.3 - Matched filter detection of a random code.

Fig. 1.4 - Matched filter detection of an optimal noise code.

I will consider any undesired signal which might be mistaken for a true signal as noise. These two features will be discussed in some detail in the true digital intrapulse modulation schemes to be discussed later.

The scheme just considered is somewhat inefficient since seven units of time are required to radiate three units of energy. If in the transmitted signal all the spaces with zero energy output in the pulse envelope had also had an output, seven units of energy would have been radiated in seven units of time. One might use some sort of frequency shift or phase modulation to accomplish this purpose, still utilizing of course the same output tube. Consider the same waveform as before with the units of energy coded as plus and minus by the modulator by means of phase shift modulation of either zero or 180° . The waveform is as shown in Figure 1.3-a. Upon reception this waveform is now fed into a delay line in the receiver as shown in Figure 1.3-b. "A" represents a straight-through amplifier. Figure 1.3-c shows the output of the delay line summer with respect to time. The output energy is now seven times that of a single pulse and has the frequency content, and therefore the range resolution, of one of the single pulses. A time delay in the output is again apparent, as are a number of noisy sub-peaks. This was a code picked at random and the results were good. There are certain classes of codes which are optimal, having even smaller noisy sub-peaks. These codes are of length $2^n - 1$ and have been studied extensively.^{5, 6, 7} They are known as L codes or pseudo-random noise codes. An example of one of these pseudo-random noise codes is - + - - + + +, where the minuses and pluses indicate the phase of the RF signal. When this particular noise code is impressed upon its matched filter, Figure 1.4-b, the output of this summed delay line is as shown in Figure 1.4-c. It is to be noted that the maximum height of the



	A	1	0	-1	0	3	0	1	2	-1	0	3	0	1	0	10
SUM	B ⁺	-1	0	-1	0	-3	0	1	8	1	0	-3	0	-1	0	10
		0	0	0	0	0	0	0	16	0	0	0	0	0	0	0

g

Matched filter detection of a complementary code of length 8.

FIGURE 1.5

"hash" (sub-peaks) is one, with a main peak of seven, whereas in the previous example picked at random the hash had a height of two. Elaspas has made a study and design of a radar utilizing this form of modulation and matched filter detection.⁸ He infers that codes of reasonable length should be used. For example, a code of length 127 would be a reasonable length to obtain the benefits of a pseudo-random matched filter radar. An optimum code of this length has a maximum subpeak of 13, which gives a good coding signal-to-noise ratio.

There is a way, however, to completely eliminate the hash, although this required the transmission of two separate sequences. Both of these sequences could be transmitted through one output tube with a moderate band width. These codes which eliminate the hash are known as complementary codes.⁹ An example of a complementary code pair being detected by a matched filter pair is shown in Figure 1.5. The two codes being transmitted will be called A and B for convenience, with their respective matched filters also being labeled in a similar fashion. The receiver output, after the complementary codes are detected by the matched filter and their outputs summed, is shown in Figure 1.5-g. This demonstrates, as was mentioned before, that the hash level is zero. This lack of hash is indicative of an infinite coding signal-to-noise ratio. This particular feature is the prime advantage of complementary sequence pairs over other forms of noise modulation. Figure 1.5-g shows the same amount of delay as did all the other forms of matched filter detectors, and shows that the sum of all the energy sub-pulses is incorporated into the main pulse at exact match.

Each of the systems described in this introduction has had the characteristic that the transmitted signal is spread over a wide frequency spectrum. This has in itself two advantages: One is that enemy detection of the signal becomes more difficult, and the other is that even after detection the jamming of a wide spectrum noise-like signal becomes

extremely difficult. Since the characteristics just described are desirable for a secure communication system as well as for a radar system, all of the advantages that accrue for the complementary sequence modulation in radar would apply in a communications scheme.

The initial major objectives of this investigation were two in number.

1. The formulation of a set of operations on complementary sequences which form a group, along with a theoretical study of the invariant properties of such a group.
2. The search for new complementary codes. An exhaustive computer search for a new code of length 26 was made as well as a partial search for codes of length 34.

During the investigation two more major objectives were added to the work.

These were:

3. The application of Hamming distance to both the complementary sequences and their group formulation.¹⁰
4. The application of Hamming vectors to complementary sequence pairs and also to their group formulation.

The original objectives were oriented toward the generation of new codes, while the last two objectives were oriented toward the decomposition of codes. All of these objectives were attained and constitute a large portion of the rest of this work. The next chapter is a study of complementary sequences designed to bring the reader to the state of the art, and contains some proofs which have not appeared elsewhere.^{1,2,9,11,12} Chapter 3 is the formulation of the operations group on complementary sequences and, along with several theorems and proofs it also contains the identification of this operations

group with a known group. Chapter 3 as written requires only a minimum of group theory knowledge. Chapters 4 and 5 have to do with the Hamming distance and Hamming vectors of complementary sequences both in standard form and in group form. Chapter 6 is a short chapter and is concerned with the proof of two theorems used in the searching for new codes. Chapters 7 and 8 describe the computer search for new kernels of length 26 and 34 respectively, the actual computer programs used being listed in the appendices. The last chapter, Number 9, contains the summary, conclusions, and suggestions for future research in this interesting field.

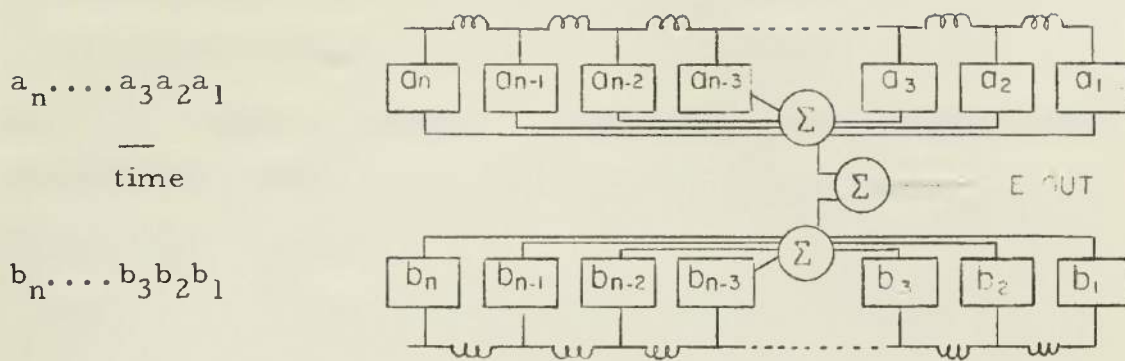
CHAPTER II

COMPLEMENTARY SEQUENCES

This chapter is presented to provide a background in complementary sequences to enable the reader to better understand the material in later chapters. Most of the material in this chapter is an adaptation of Golay's "Complementary Series" with expansions and deletions to fit the needs of this paper.⁹ The formalization of some proofs and the addition of new proofs given here have not appeared in print elsewhere; however, it is obvious from the tone of Golay's work that he was aware of these proofs. This modification of Golay's work is offered for completeness of the dissertation and not as new work.

In the introduction a pair of codes of length n were discussed which had the characteristic that when detected by matched filters, the sum of the two filter outputs was everywhere zero, except where the signal patterns had zero time delay with reference to the filter patterns. At the zero pattern delay time the output was $2n$, leading to an infinite coding signal-to-noise ratio. A study of the necessary and sufficient conditions for this to be true will now be considered.

Assume a complementary code pair (A, B) and suppose that one of the pair, i. e. A , is longer than the other. If this be true then there will be an output from the A filter with no cancelling output from the B filter. This is not allowed; therefore the A and B codes must be of equal length. Therefore assume a code A , n bits in length, with elements a_1 through a_n , where each a_i is either $+1$ or -1 . Also, assume a second code B , n bits in length, with elements b_i similarly defined. Detect each of these codes with a matched filter (the time inverse of the code) and then sum the outputs of the A filter and the B filter. At time 1 (see Figure 2.1) the signals a_1 and b_1 have entered the filter segments a_n and b_n respectively. Since at this



The a_i and b_i of the filter are either ± 1 indicating straight thru amplifiers or inverters.

Time 1 $a_4 a_3 a_2 a_1$ signal
 $a_n a_{n-1} a_{n-2} \dots a_1$ filter

$b_3 b_2 b_1$ signal
 $b_n b_{n-1} b_{n-2} \dots b_1$ filter

Time 2 $a_4 a_3 a_2 a_1$
 $a_n a_{n-1} a_{n-2} \dots$

$b_4 b_3 b_2 b_1$
 $b_n b_{n-1} b_{n-2}$

Time r $a_r \dots a_1$
 $a_n \dots a_{n+1-r} a_{n+1-(r+1)} \dots$

Matched filter detection of a complementary pair of length n .

FIGURE 2.1

time the delay between signal pattern and filter pattern is not zero, the summed output voltage must be zero. This gives the condition $a_1 a_n + b_1 b_n = 0$. At time 2 (Figure 2.1) the signals are one segment farther into the filter and the output must again be zero, giving $a_1 a_{n-1} + a_2 a_n + b_1 b_{n-1} + b_2 b_n = 0$.

At time 3: $a_1 a_{n-2} + a_2 a_{n-1} + a_3 a_n + b_1 b_{n-2} + b_2 b_{n-1} + b_3 b_n = 0$

At time r ($r \neq n$): $a_1 a_{n+1-r} + a_2 a_{n+1-(r-1)} + \dots + a_r a_n + b_1 b_{n+1-r} + \dots + b_r b_n = 0$

At time n : $a_1 a_1 + a_2 a_2 + \dots + a_n a_n + b_1 b_1 + \dots + b_n b_n = 2n$

for at time n the pattern of the signal must exactly match the pattern of the filter. The explicit statement of the conditions given above for all times is

$$F_j = \sum_{i=1}^{i=n-j} a_i a_{i+j} + \sum_{i=1}^{i=n-j} b_i b_{i+j} = \begin{cases} 0 & j \neq 0 \\ 2n & j = 0 \end{cases} \quad (2.1)$$

An examination of the equation at time 1 ($j = n - 1$) shows the following eight possible solutions:

a_1	a_n	b_1	b_n
1	1	1	-1
1	1	-1	1
-1	-1	1	-1
-1	-1	-1	1
-1	1	1	1
1	-1	1	1
-1	1	-1	-1
1	-1	-1	-1

These possibilities show that if the A pair are alike, the B pair must be unlike, and if the A pair are unlike, the B pair must be alike. The equation at time 2 ($j = n - 2$) indicates that if the number of likes (like pairs, i.e. a_1 and a_{n-1}) of the A code and the number of unlikes (unlike pairs) of the B code are equal, and if the number of unlikes of A equals the number of likes of B, the equation is satisfied. Extending

this reasoning to time r shows again that if the likes of A equal the unlikes of B and if the unlikes of A equal the likes of B the equation is again satisfied.

It will now be proved that the last half of the previous statement is not necessary, since the first half of the statement contains the necessary and sufficient requirement.

Let U_a = number of unlike pairs in A for the spacing specified.
(j in equation 2.1)

Let L_a = number of like pairs in A for the spacing specified.
(j in equation 2.1)

Let U_b = number of unlike pairs in B for the spacing specified.
(j in equation 2.1)

Let L_b = number of like pairs in B for the spacing specified.
(j in equation 2.1)

Assume that $L_a = U_b$, we want to prove that $U_a = L_b$.

Since the sum of the number of like pairs at each spacing with the number of unlike pairs at the same spacing must equal the total number of possibilities,

$$U_a + L_a = n - (n-r) \quad \text{for } r \neq n \text{ where } n \text{ is the length of the code} \\ \text{and } (n-r) \text{ is the number of units of time delay} \\ \text{from exact match.}$$

For the same reason

$$U_b + L_b = n - (n-r).$$

But since $U_b = L_a$,

$$U_a + U_b = n - (n-r).$$

Therefore $U_a = L_b$.

The definition of a complementary pair of sequences will now be given. A pair of binary sequences of equal length with the number of like pairs of one sequence equal to the number of unlike pairs of the other sequence for each possible spacing is said to be a complementary code.

Two schemes of representation will be used in this paper. These schemes will be used interchangeably at the convenience of the author. The first representation is the one already presented where each a_i and b_i is either $+1$ or -1 . The operation used in this case is ordinary multiplication. The second representation will use the operation of modulo two addition denoted by \oplus , where the elements a_i and b_i are either 0 or 1. That these two groups $(1, -1; \cdot)$ and $(0, 1; \oplus)$ are isomorphic is shown in the following tables.

\cdot	1	-1
1	1	-1
-1	-1	1

\oplus	0	1
0	0	1
1	1	0

$$\begin{aligned} \text{where } 1 &\rightarrow 0 \\ -1 &\rightarrow 1 \end{aligned}$$

The first form will normally be used when correlating of the codes with their matched filter is under discussion while the second form will normally be used when discussing the intrinsic properties of the individual codes.

The necessary and sufficient condition for a pair of sequences to be complementary, is in terms of the modulo two representation

$$F_j = \sum_{i=1}^{i=n-j} (a_i \cdot a_{i+j}) = \sum_{i=1}^{i=n-j} (b_i \oplus b_{i+j} \oplus 1) \text{ for all } j, 1 \leq j \leq n-1 \quad (2.2)$$

or an equivalent formulation by change of summation index

$$f_j = \sum_{i=1}^{i=j} (a_i \cdot a_{n-j+i}) = \sum_{i=1}^{i=j} (b_i \oplus b_{n-j+i} \oplus 1) \quad (2.3)$$

The above Σ is standard summation. If modulo two summation is desired the symbol \oplus will be utilized. (The equivalent conditions are given since they are both used in formal proofs in a later portion of the paper. Due to the possibility of confusion it seems more appropriate

to introduce the second form now rather than later when needed.)

$U_a = L_b$ has been used to represent these same equations, and will be used elsewhere in the paper when it is more convenient to use it.

An interesting property of any system of codes is the possible invariances it may have under various types of transformations. It therefore seems appropriate to consider the possible transformations on a complementary sequence pair which leave the pair complementary and of the same length after the transformation. A necessary tool for this study is the property that $a_i \oplus a_{i+j} = \bar{a}_i \oplus \bar{a}_{i+j}$ where the symbol \bar{a}_i means the complement of a_i , $\bar{0} = 1$, $\bar{1} = 0$.

Theorem 2.1

The modulo two sum of a pair of binary numbers is equal to the modulo two sum of their complements.

The truth of the theorem is obvious from the following exhaustive table of four possibilities:

$a_i \oplus a_{i+j}$	$\bar{a}_i \oplus \bar{a}_{i+j}$
$1 \oplus 1 = 0$	$\bar{1} \oplus \bar{1} = 0 \oplus 0 = 0$
$1 \oplus 0 = 1$	$\bar{1} \oplus \bar{0} = 0 \oplus 1 = 1$
$0 \oplus 1 = 1$	$\bar{0} \oplus \bar{1} = 1 \oplus 0 = 1$
$0 \oplus 0 = 0$	$\bar{0} \oplus \bar{0} = 1 \oplus 1 = 0$

Therefore $a_i \oplus a_{i+j} = \bar{a}_i \oplus \bar{a}_{i+j}$.

* * *

In each of the following proofs it will be considered that A and B are a complementary sequence pair of length n. Complementing each element in a code is called complementing the code.

Theorem 2.2

Complementing the A code, or the B code, or both the A and B codes, results in a pair of complementary codes. (The proof is given just for the A code but is identical for the B code and

from both of these for the A and B codes.)

By hypothesis

$$F_j = \sum_{i=1}^{i=n-j} (a_i \oplus a_{i+j}) = \sum_{i=1}^{i=n-j} (b_i \oplus b_{i+j} \oplus 1).$$

Complementing the A code transforms

$$a_i \rightarrow \bar{a}_i,$$

but by Theorem 2.1

$$a_i \oplus a_{i+j} = \bar{a}_i \oplus \bar{a}_{i+j}.$$

Upon substitution

$$F_j = \sum_{i=1}^{i=n-j} (\bar{a}_i \oplus \bar{a}_{i+j}) = \sum_{i=1}^{i=n-j} (b_i \oplus b_{i+j} \oplus 1).$$

Therefore complementing the A code has no effect on complementarity, and similarly for the B code or for both codes.

* * *

Interchanging the first and last bits of a code, the second and next to last bits, the third and third from last bits, and so on is called time inverting or time reversing a code.

Theorem 2.3

Time inverting the A code, or the B code, or both the A and B codes, results in a complementary pair. (This proof is again just for the A code, but also applies to the B code and therefore to both the A and B codes.)

By hypothesis

$$f_j = \sum_{i=1}^{i=n-j} (a_i \oplus a_{i+j}) = \sum_{i=1}^{i=n-j} (b_i \oplus b_{i+j} \oplus 1).$$

The time inverse of A causes each

$$a_i \rightarrow a_{n+1-i}.$$

Applying this time reversal just to the A code shows that

$$F_j = \sum_{i=1}^{i=n-j} (a_i \oplus a_{i+j}) \rightarrow \sum_{i=1}^{i=n-j} (a_{n+1-i} \oplus a_{n+1-(i+j)})$$

Expanding both expressions for a few terms and remembering that modulo 2 addition is commutative as is regular addition we see that

(on the left)

(on the right)

$$\text{for } j=n-1, F_{n-1} = a_1 \oplus a_n$$

$$F_{n-1} = a_n \oplus a_1$$

$$\text{for } j=n-2, F_{n-2} = a_1 \oplus a_{n-1} + a_2 \oplus a_n \quad F_{n-2} = a_n \oplus a_2 + a_{n-1} \oplus a_1$$

These expressions indicate that by a change in summation index the two forms of F_j are equal, as is the case. Therefore time reversing the A code has no effect on the complementary property, and similarly for the B code and for both the A and B codes.

* * *

The altering of a code is a transformation where every other bit of both codes is complemented. In a different portion of the paper a distinction is made between altering odd bits and altering even bits. In the following proof, however, just the even bits are altered, although a slightly modified proof would hold for altering the odd bits.

Theorem 2.4

The result of altering a complementary pair of sequences is again a complementary sequence pair.

Since the codes are a complementary pair,

$$F_j = \sum_{i=1}^{i=n-j} (a_i \oplus a_{i+j}) = \sum_{i=1}^{i=n-j} (b_i \oplus b_{i+j} \oplus 1)$$

Divide the F_j into two groups, F_k for $j =$ an even number and F_r for $j =$ an odd number. The portion of the above equations for $j =$ an even number is

$$F_k = \sum_{i=1}^{i=n-k} (a_i \oplus a_{i+k}) = \sum_{i=1}^{i=n-k} (b_i \oplus b_{i+k} \oplus 1)$$

For the theorem to be true, when j is an even number the above set of equations F_k must be satisfied for all k when the pair is altered. (The proof is given only in terms of A , but duplicate steps must also be applied to B .)

When i is an even number, then $i+k$ is an even number. In the altering process since all even bits are complemented $a_i \oplus a_{i+k} \rightarrow \bar{a}_i \oplus \bar{a}_{i+k}$, but by Theorem 2.1 these are equal. When i is an odd number, then $i+k$ is an even number and these are not affected by the altering process, since only even bits are complemented. Therefore F_k is invariant under altering, or symbolically $F_k = F_{kalt}$.

Examination of F_r where r is an odd value of j , shows that of the pair a_i, a_{i+r} one of the bits is complemented and the other is unchanged.

$$a_i \oplus a_{i+r} \rightarrow (\bar{a}_i \oplus a_{i+r} \text{ or } a_i \oplus \bar{a}_{i+r}) = \overline{a_i \oplus a_{i+r}}.$$

Therefore $F_{ralt} = (n-1) - F_r$ for both the A code and the B code. This signifies the likes of the A code for odd spacings are changed into unlikes, and the unlikes of the A code for odd spacings are changed into likes; however, for each change in A there is an opposite change in B so the total likes in A still equals the total unlikes in B for each spacing. It is to be noted that altering both the even and the odd bits is the same as complementing both codes.

* * *

The operation of interchanging the A and B codes, although trivial, is also a transformation. There are a total of sixty four possible transformations. These can all be generated by combining the above listed transformations. They will be discussed much more thoroughly in Chapter 3.

Whenever a new facet of a pair of sequences such as the complementary property is discovered, it is natural to wonder just how general this property might be. One question of interest would be the possible length limitations of the code pairs that might be forced by the complementary property. The following proof due to Golay shows that complementary sequences must have a length n of the form $n = x^2 + y^2$, where n is an even integer and x and y are integers. The first theorem to be proved demonstrates that n must be an even integer and the second theorem shows the $x^2 + y^2$ form.

Theorem 2.5

A necessary condition for a sequence pair to be complementary is that their length be an even number.

By hypothesis

$$F_j = \sum_{i=1}^{i=n-j} a_i \oplus a_{i+j} = \sum_{i=1}^{i=n-j} b_i \oplus b_{i+j} \oplus 1.$$

Let $G_j = F_j$ modulo 2, so that

$$G_j = \sum_{i=1}^{i=n-j} a_i \oplus a_{i+j} = \sum_{i=1}^{i=n-j} b_i \oplus b_{i+j} \oplus 1.$$

An expansion of G for a few different values of j will suggest the proof. Start with $j = n-1$.

$$G_{n-1} = a_1 \oplus a_n = b_1 \oplus b_n \oplus 1,$$

$$G_{n-2} = a_1 \oplus a_{n-1} \oplus a_2 \oplus a_n = b_1 \oplus b_{n-1} \oplus 1 \oplus b_2 \oplus b_n \oplus 1,$$

$$G_{n-3} = a_1 \oplus a_{n-2} \oplus a_{n-1} \oplus a_n \oplus a_3 \oplus a_n = b_1 \oplus b_{n-2} \oplus 1 \oplus b_{n-1} \oplus 1 \oplus b_3 \oplus b_n \oplus 1.$$

Change the order of the sums in G_{n-2} and G_{n-3} to achieve the following:

$$G_{n-2} = a_1 \oplus a_2 \oplus a_{n-1} \oplus a_n = b_1 \oplus b_2 \oplus b_{n-1} \oplus b_n \oplus \sum_{i=1}^2 1,$$

$$G_{n-3} = a_1 \oplus a_2 \oplus a_3 \oplus a_{n-2} \oplus a_{n-1} \oplus a_n = b_1 \oplus b_2 \oplus b_3 \oplus b_{n-2} \oplus b_{n-1} \oplus b_n \oplus \sum_{i=1}^3 1.$$

In general for $j = n-r$

$$G_{n-r} = a_1 \oplus a_2 \oplus \dots \oplus a_r \oplus a_{n-(r+1)} \oplus a_{n-(r+2)} \oplus \dots \oplus a_{n-1} \oplus a_n = b_1 \oplus b_2 \oplus \dots \oplus$$

$$\dots \oplus b_r \oplus b_{n-(r+1)} \oplus b_{n-(r+2)} \oplus \dots \oplus b_{n-1} \oplus b_n \oplus \sum_{l=1}^r 1,$$

and for $j=n-(r+1)$

$$G_{n-(r+1)} = a_1 \oplus a_2 \oplus \dots \oplus a_r \oplus a_{r+1} \oplus a_{n-r} \oplus a_{n-(r+1)} \oplus \dots \oplus a_{n-1} \oplus a_n = b_1 \oplus b_2 \oplus \dots \oplus b_r \oplus b_{r+1} \oplus b_{n-r} \oplus b_{n-(r+1)} \oplus b_{n-(r+2)} \oplus \dots \oplus b_{n-1} \oplus b_n \oplus \sum_{l=1}^{r+1} 1.$$

Forming the sum $G_{n-r} \oplus G_{n-(r+1)}$ will leave only five terms in the equation since G_{n-r} and $G_{n-(r+1)}$ differ only in a_{n-r} , a_{r+1} , b_{r+1} , and a , because

$$\sum_{l=1}^r 1 \oplus \sum_{l=1}^{r+1} 1 = 1.$$

$$G_{n-r} \oplus G_{n-(r+1)} = a_{r+1} \oplus a_{n-r} = b_{r+1} \oplus b_{n-r} \oplus 1.$$

Adding $b_{r+1} \oplus b_{n-r}$ modulo 2 to both sides of the equation gives

$$a_{r+1} \oplus a_{n-r} \oplus b_{r+1} \oplus b_{n-r} = 1. \quad (2.4)$$

Let $n=2s-1$ and $r=s-1$; then to satisfy the equation just derived

$$a_s \oplus a_{2s-1-s+1} \oplus b_s \oplus b_{2s-1-s+1} = a_s \oplus a_s \oplus b_s \oplus b_s = 1, \text{ which}$$

is obviously false.

Therefore n cannot have the form $2s-1$ and must be an even number.

* * *

Equation 2.4 which was a step in the previous proof, is a very important necessary condition for a pair of codes to be complementary and will be referred to in the future as the parity check. The next theorem which shows the $n=x^2+y^2$ form, starts with the basic assumption that A and B are a complementary pair of length n .

Theorem 2.6

A necessary condition for a pair of codes to be complementary is that their length be the sum of the squares of two integers.

By hypothesis

$$F_j = \sum_{i=1}^{i=n-j} (a_i \oplus a_{i+j} \oplus 1) = \sum_{i=1}^{i=n-j} (b_i \oplus b_{i+j}).$$

Assume that the A code satisfies F_j and that the A sequence contains

p ones and n-p zeros. The A code has a weight of p. Similarly assume that the B code satisfies F_j and that its sequence contains q ones and n-q zeros. The B code has a weight of q. For the total likes in the A code, each bit is matched against each other bit exactly once. For the total unlikes in the B code each bit is again matched against each other bit once. There are two possibilities of likes in the A code, 1 matched against 1, and 0 matched against 0. The total number of like pairs for the A code is therefore the number of combinations of p ones taken 2 at a time plus the number of combinations of n-p zeros taken 2 at a time, for the unlike pairs in the B code, q ones are matched against n-q zeros.

This gives the equation

$$\frac{p(p-1)}{2} + \frac{(n-p)(n-p-1)}{2} = q(n-q)$$

expanding and simplifying

$$n = n^2 - 2qn + 2q^2 + 2p^2 - 2np,$$

adding and subtracting 2pq gives

$$n = n^2 - 2qn + 2q^2 + 2p^2 - 2np + 2qp - 2qp,$$

collecting terms gives

$$n = n^2 + q^2 + p^2 + 2qp - 2np - 2nq + p^2 - 2pq + q^2$$

combining terms gives

$$n = (n-p-q)^2 + (p-q)^2 \quad (2.5)$$

Therefore complementary sequence lengths are permissible only in lengths which are even numbers and formed by the sum of two squared integers.

* * *

A list of all possible code lengths up to 200 with the number of ones allowed in the A and B codes is given in Table 2.1.

*Indicates possible kernel

<u>Code Lengths</u>	<u>Unordered Weights of (A, B)</u>	<u>Code Lengths</u>	<u>Unordered Weights of (A, B)</u>
*2	(2, 1)(1, 0)	104	(58, 56)(58, 48)(48, 46)(56, 46)
4	(3, 3)(1, 1)(3, 1)	*106	(60, 55)(51, 46)(60, 51)(55, 46)
8	(6, 4)(4, 2)	116	(65, 61)(55, 51)(65, 55)(61, 51)
*10	(7, 6)(4, 3)(7, 4)(6, 3)	*122	(67, 66)(56, 55)(67, 56)(66, 55)
16	(10, 10)(6, 6)(10, 6)	128	(72, 64)(64, 56)
*18	(12, 9)(9, 6)	*130	(72, 69)(72, 61)(61, 58)(64, 57)
20	(13, 11)(9, 7)(11, 7)		(73, 66)(73, 64)(64, 57)(66, 57)
*26	(16, 15)(11, 10)	136	(76, 70)(66, 60)(76, 66)(70, 60)
	(15, 10)(16, 11)	144	(78, 78)(66, 66)(78, 66)
32	(20, 16)(16, 12)	*146	(81, 76)(70, 65)(81, 70)(76, 65)
*34	(21, 18)(16, 13)	148	(81, 79)(69, 67)(81, 69)(79, 67)
	(21, 16)(18, 13)	160	(88, 84)(76, 72)(88, 76)(84, 72)
36	(21, 21)(15, 15)(21, 15)	*162	(90, 81)(81, 72)
40	(24, 22)(18, 16)	164	(91, 83)(81, 73)(91, 81)(83, 73)
	(24, 18)(22, 16)	*170	(94, 87)(83, 76)(94, 83)(87, 76)
*50	(30, 25)(25, 20)(29, 28)		(92, 91)(79, 78)(92, 79)(91, 78)
	(29, 22)(28, 21)(22, 21)	*178	(97, 94)(97, 84)(84, 81)(94, 81)
52	(31, 27)(31, 25)		
	(25, 21)(27, 21)	180	(99, 93)(87, 81)(99, 87)(93, 81)
*58	(34, 31)(27, 24)		
	(34, 27)(31, 24)	*194	(106, 101)(93, 88)
64	(36, 28)(36, 36)(28, 28)		(106, 93)(101, 88)
68	(39, 37)(31, 29)	196	(105, 105)(91, 91)(105, 91)
	(39, 31)(37, 29)	200	(110, 100)(108, 106)(100, 90)
72	(42, 36)(36, 30)		(94, 92)(94, 108)(106, 92)
*74	(43, 38)(36, 31)		
	(43, 36)(38, 31)		
80	(46, 42)(38, 34)	All possible complementary code lengths up to 200 with unordered possible weights.	
	(46, 38)(42, 34)		
*82	(46, 45)(37, 36)		
	(46, 37)(45, 36)		
*90	(51, 48)((51, 42)		
	(42, 39)(48, 39)		
*98	(56, 49)(49, 42)		
100	(55, 55)(55, 45)(45, 45)		
	(57, 51)(57, 49)(51, 43)		
	(49, 43)		

TABLE 2.1

The shortest possible complementary pair is $A=11$, $B=10$. This pair or any of its transformations is called a kernel of length two, or the quad. A kernel is a basic length code which cannot be decomposed into shorter length codes by an inversion of the standard generating methods to be explained later. Some possible kernel lengths are 2, 10, 18, 26, 34, 50, etc. although codes for all of these do not exist. Complementary sequences which are not kernels are called composite complementary sequences.

The above list of kernel lengths did not include $n=4$ or $n=8$ which are possible complementary sequence lengths. These if they exist must therefore be composite. All possible codes of length four can be generated from the exhaustive list of possibilities for four binary digits given in Table 2.2.

0000	1000
0001	1001
0010	1010
0011	1011
0100	1100
0101	1101
0110	1110
0111	1111

Table 2.2. All Possible Binary Numbers of Length Four.

The unordered possible pairs of ones in the codes are (1,1), (3,3) and (3,1). Thus all possible codes of length four have either one 1 or three 1's because of this limitation. This reduces the table to the eight following numbers of length four:

0001	0111
0010	1101
0100	1011
1000	1110

The possible unordered (1,1) code pairs are the following four:

1000	1000	0001	0001
0100	0010	0100	0010

The possible unordered (3,1) pairs of codes are the following eight:

0111	0111	1110	1110	1011	1011	1101	1101
0010	0100	0100	0010	0001	1000	0001	1000

The possible unordered (3,3) code pairs are

0111	0111	1110	1110
1011	1101	1011	1101

Comparing the last code to the kernel of length two, $A=11$, $B=10$, shows that it might have been constructed by writing in time sequence

$$S_1 = AB = a_1 a_2 b_1 b_2 = 1110,$$

$$S_2 = A\bar{B} = a_1 a_2 \bar{b}_1 \bar{b}_2 = 1101.$$

The next to last sequence pair might have been formed by interlacing A and B in the following manner:

$$T_1 = a_1 b_1 a_2 b_2 = 1110,$$

$$T_2 = a_1 \bar{b}_1 a_2 \bar{b}_2 = 1011.$$

All the rest of the 16 pairs could be considered either as transformations of the last pair, or as being formed by the same two composite generating operations used above on the 8 transformations of the kernel of length two. A general proof will now be given to show that both the time sequence scheme (S_1, S_2) and the interlace scheme (T_1, T_2) will always form complementary sequence pairs providing that A and B are a complementary sequence pair.

Theorem 2.7

If $A = a_1 a_2 a_3 \dots a_{n-1} a_n$

$B = b_1 b_2 b_3 \dots b_{n-1} b_n$

are a complementary sequence pair, then

$$C = a_1 a_2 \dots a_n b_1 b_2 \dots b_n$$

$$D = a_1 a_2 a_3 \dots a_n \bar{b}_1 \bar{b}_2 \dots \bar{b}_n$$

are a complementary sequence pair.

Theorem 2.8

If (A, B) are a complementary sequence pair, then

$$C = a_1 b_1 a_2 b_2 \dots a_n b_n$$

$$D = a_1 \bar{b}_1 a_2 \bar{b}_2 \dots a_n \bar{b}_n$$

form a complementary sequence pair.

The following notation will be used in the proof:

$$U_a = \sum_{i=1}^{i=n-j} a_i \oplus a_{i+j} \quad \text{for all } j, 1 \leq j \leq n-1$$

$$L_a = \sum_{i=1}^{i=n-j} a_i \oplus a_{i+j} \oplus 1 \quad \text{for all } j, 1 \leq j \leq n-1.$$

It is understood that the two CD sequences in the two theorems above are not the same but because the proofs are identical two sets of symbols will not be used. For each spacing j there are three possible ways for bits to match, within A, within B, or from a bit of A to a bit in B. These possibilities are denoted by L_a , L_b and L_{ab} or U_a , U_b , U_{ab} respectively.

The necessary and sufficient condition for C and D to be a complementary sequence pair is given by $L_c = L_a + L_b + L_{ab}$ and $U_d = U_a + U_b + U_{ab}$ where $L_c = U_d$.
Now $L_a = U_b$ and $L_b = U_a$ by definition of complementary.

Adding

$$L_a + L_b = U_a + U_b,$$

but

$$U_b = U_{\bar{b}} \quad \text{since} \quad \sum_{i=1}^{i=n-1} b_i \oplus b_{i+j} = \sum_{i=1}^{i=n-1} \bar{b}_i \oplus \bar{b}_{i+j}$$

so substituting

$$L_a + L_b = U_a + U_{\bar{b}}$$

All bit matches from A to B in the C sequence are paired in the D sequence with bit matches from A to \bar{B} therefore,

$$L_{ab} = U_{a\bar{b}} ,$$

and adding

$$\begin{array}{ccccccc} L_a + L_b + L_{ab} & = & U_a + U_{\bar{b}} + U_{a\bar{b}} & \text{or} & L_c = U_d \\ & & * & & * & & * \end{array}$$

Golay describes two other methods of generating composite sequences from shorter complementary sequences. These methods will not be used too extensively in the rest of the paper and will therefore just be mentioned rather than proved. Given two complementary sequences pairs (A,B) (C,D), A of length m and C of length n.

Let

$$\begin{aligned} U_1 &= A^{c_1} A^{c_2} \dots A^{c_n} B^{d_1} B^{d_2} \dots B^{d_n} \\ U_2 &= A^{d_n} \dots A^{d_1} B^{\bar{c}_n} \dots B^{\bar{c}_1}, \end{aligned} \quad (2.6)$$

and

$$\begin{aligned} V_1 &= A^{c_1} B^{d_1} \dots A^{c_n} B^{d_n} \\ V_2 &= A^{d_n} B^{\bar{c}_n} \dots A^{d_1} B^{\bar{c}_1}. \end{aligned} \quad (2.7)$$

It can be shown that the pairs of U are complementary as are the pairs of V. Where, if an exponent is a one the A or B code is left unchanged, and if the exponent is a zero the A or B code is complemented. The lengths of codes which can be generated from these two methods are 2mn where m is the length of the (A,B) sequence pair and n is the length of the (C,D) sequence pair. There is also another special method of generation which applies only to codes of length 2^r . This method is explained in Chapter 5 where it is necessary for the completeness of a proof.

The kernel of length 2, or the quad, has been discussed in the previous pages. There are two kernels of length 10. These are 1001010001 and 1000000110, and a second kernel 0101000011 and 0000100110. The next possible kernel size is 18 and Golay has proved

by exhaustive search that no kernel of this length exists. Kruskal has since then completed the proof analytically.¹³ One of the goals of the writer in his study was to make an exhaustive search for codes of kernel length 26. This search disclosed there was only one kernel of length 26. Chapter 7 of this paper describes this search in detail and Chapter 8 describes attempts to find kernels of length 34.

CHAPTER III

THE GROUP OF OPERATIONS

A large portion of this investigation is devoted to the set of allowable transformations on complementary sequences. These allowable transformations possess the characteristic that the length of the sequences and the complementary property are invariant under transformation. To be useful in searching for new kernels it is also desirable that these transformations have the following features:

1. That they form a group for all general transformations. Where a general transformation is one which can be applied to any complementary code pair.
2. The transformations as represented be simple manipulations on the elements of the group.
3. The supplementary characteristic (to be explained in Chapter 6) be a property of the representation.
4. The parity check be easily made.

Some of these transformation operations were discussed, and the proof of invariance given for all group generators, in the background chapter, Chapter 2, on complementary sequences. As a continuation, a brief review of the transformation operations and a tagging of symbols to these operations will prove useful. Each basic operation symbol and its definition is given in Table 3.1. More definitions are given than are necessary to generate the group; however, the redundancy seems to the writer to be an aid to understanding.

A check through Table 3.1 shows that the transformations affect the complementary sequence pair in four different ways. C_1 , C_2 , T_1 , T_2 affect all the bits of one of the code pair. E , T , C affect all bits of both codes, while A_1 and A_2 affect half of the bits of both codes. The identity, I , of course has no affect on either code. A reasonable way to form the elements of the group to account for the above affects would be to divide each code of the pair into two units. These would be A_e (even bits) and A_o (odd bits), also B_e (even bits) and B_o (odd bits).

I The Identity

$$a_i \rightarrow a_i$$

$$b_i \rightarrow b_i$$

T₁ A code time reversal

$$a_i \rightarrow a_{n+1-i}$$

$$b_i \rightarrow b_i$$

T₂ B code time reversal

$$a_i \rightarrow a_i$$

$$b_i \rightarrow b_{n+1-i}$$

T Time reversal of both codes

$$a_i \rightarrow a_{n+1-i}$$

$$b_i \rightarrow b_{n+1-i}$$

A₁ Alter odd bits

$$a_{2i-1} \rightarrow \bar{a}_{2i-1} \quad a_{2i} \rightarrow a_{2i}$$

$$b_{2i-1} \rightarrow \bar{b}_{2i-1} \quad b_{2i} \rightarrow b_{2i}$$

A₂ Alter even bits

$$a_{2i-1} \rightarrow a_{2i-1} \quad a_{2i} \rightarrow \bar{a}_{2i}$$

$$b_{2i-1} \rightarrow b_{2i-1} \quad b_{2i} \rightarrow \bar{b}_{2i}$$

C₁ Complement the A code

$$a_i \rightarrow \bar{a}_i$$

$$b_i \rightarrow b_i$$

C₂ Complement the B code

$$a_i \rightarrow a_i$$

$$b_i \rightarrow \bar{b}_i$$

C Complement both codes

$$a_i \rightarrow \bar{a}_i$$

$$b_i \rightarrow \bar{b}_i$$

Some operations.

E Exchange the A code with the B code

$$a_i \rightarrow b_i$$

$$b_i \rightarrow a_i$$

$$A_o = a_1 a_3 a_5 \dots a_{n-1}$$

$$A_e = a_2 a_4 a_6 \dots a_n$$

$$B_o = b_1 b_3 b_5 \dots b_{n-1}$$

$$B_e = b_2 b_4 b_6 \dots b_n$$

However, this particular division for representing the complementary sequences proves to be very awkward for the time reversal operations (T_1, T_2, T) and for the ease of the parity check condition.

Fortunately, one simple change, that of time reversing the second and fourth row (even bits of A and B), clears up these difficulties.¹⁴

The method for converting from a complementary sequence pair into the standard group form is therefore

$$\begin{aligned} A &= a_1 a_2 a_3 \dots a_{n-1} a_n \begin{cases} I = a_1 a_3 a_5 \dots a_{n-1} \\ II = a_n a_{n-2} \dots a_2 \end{cases} \\ B &= b_1 b_2 b_3 \dots b_{n-1} b_n \begin{cases} III = b_1 b_3 b_5 \dots b_{n-1} \\ IV = b_n b_{n-2} \dots b_2 \end{cases} \end{aligned} \quad (3.1)$$

If the matrix (I II III IV) is taken as the symbol for the above grouping the Identity operation, I, should yield (I II III IV). Post multiplying (I II III IV)

$$\text{by } \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ gives (I II III IV), therefore I can be taken}$$

as the identity matrix of rank 4, using the operation matrix multiplication on the right. Similarly T_1 has the effect of exchanging the first and second columns yielding (II I III IV). Now post multiplying

(I II III IV) by

$$\begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \text{ yields (II I III IV) the desired result so the}$$

matrix just used could represent T_1 . Similarly,

$$T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad C_1 = \begin{bmatrix} \bar{1} & 0 & 0 & 0 \\ 0 & \bar{1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad C_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \bar{1} & 0 \\ 0 & 0 & 0 & \bar{1} \end{bmatrix}$$

$$A_1 = \begin{bmatrix} \bar{1} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \bar{1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad A_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \bar{1} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \bar{1} \end{bmatrix} \quad E = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Matricies are often a very convenient method of expressing transformations. However, when the elements are few and the numbers in them simple it is often more convenient to express the operation differently. For instance (I II III IV) operated upon by T_1 yields (II I III IV). This could be just as well expressed as

$$T_1 = \begin{matrix} \text{I} & \text{II} & \text{III} & \text{IV} \\ \text{II} & \text{I} & \text{III} & \text{IV} \end{matrix} .$$

This indicates that the top row is transformed into the bottom row by the operation, or in even more shortened form, $T_1 = (\text{II I III IV})$ with the top row understood which means, as is common practice, $T_1 (\text{I II III IV}) = (\text{II I III IV})$. Another example is $A_1 = (\bar{\text{I}} \text{ II } \overline{\text{III}} \text{ IV})$, [which means $A_1 (\text{I II III III}) = (\bar{\text{I}} \text{ II } \overline{\text{III}} \text{ IV})$] where the bar across I and III indicates the negative element in the matrix or in actuality the complement of the portion of the code contained in I and III. Table 3.2 is a list of all 32 transforms generated from Table 3.1 both by symbol and group element representation. There are actually 64 transform operations in the entire group. However, the exchange operation (III IV I II) is not considered along with all the 32 elements it would generate. In this paper the pair (I II) and (III IV) are generally considered as being an unordered pair, which eliminates the exchange operation.

The second feature listed under desirable characteristics, that of simple manipulation of the elements of the group is satisfied.

The supplementary property which is discussed in Chapter 6 is also satisfied by the formulation of groups from complementary

sequences by this method. The parity check, $a_i \oplus b_i \oplus a_{n+1-i} \oplus b_{n+1-i} = 1$, is also satisfied under all transforms and is easily checked since each column of equation 3.1 is one solution of the parity check.

$$\begin{aligned}
 I &= a_1 a_3 a_5 \dots a_i \dots a_{n-1} \\
 II &= a_n a_{n-2} \dots a_{n+1-i} a_2 \\
 III &= b_1 b_3 b_5 \dots b_i \dots b_{n-1} \\
 IV &= b_n b_{n-2} \dots b_{n+1-i} b_2
 \end{aligned} \tag{3.1}$$

$I = (I \ II \ III \ IV)$	$R = (I \ \overline{II} \ IV \ \overline{III})$
$A_1 = (\overline{I} \ II \ \overline{III} \ IV)$	$Q = (II \ \overline{I} \ IV \ \overline{III})$
$A_2 = (I \ \overline{II} \ III \ \overline{IV})$	$P = (\overline{II} \ I \ \overline{IV} \ III)$
$T_1 = (II \ I \ III \ IV)$	$O = (\overline{II} \ \overline{I} \ III \ IV)$
$T_2 = (I \ II \ IV \ III)$	$N = (II \ I \ \overline{III} \ \overline{IV})$
$T = (II \ I \ IV \ III)$	$M = (\overline{I} \ \overline{II} \ IV \ III)$
$C_1 = (\overline{I} \ \overline{II} \ III \ IV)$	$L = (\overline{II} \ \overline{I} \ \overline{III} \ \overline{IV})$
$C_2 = (I \ II \ \overline{III} \ \overline{IV})$	$K = (\overline{I} \ \overline{II} \ \overline{IV} \ \overline{III})$
$C = (\overline{I} \ \overline{II} \ \overline{III} \ \overline{IV})$	$J = (\overline{II} \ \overline{I} \ \overline{IV} \ \overline{III})$
$Z = (II \ \overline{I} \ \overline{III} \ IV)$	$H = (I \ II \ \overline{IV} \ \overline{III})$
$Y = (\overline{II} \ I \ \overline{III} \ IV)$	$G = (\overline{II} \ \overline{I} \ IV \ III)$
$X = (\overline{II} \ I \ III \ \overline{IV})$	$F = (I \ \overline{II} \ \overline{III} \ IV)$
$W = (II \ \overline{I} \ III \ \overline{IV})$	$D = (\overline{I} \ II \ III \ \overline{IV})$
$V = (\overline{I} \ II \ IV \ \overline{III})$	$\emptyset = (II \ I \ \overline{IV} \ \overline{III})$
$U = (I \ II \ \overline{IV} \ III)$	$B = (\overline{II} \ I \ IV \ \overline{III})$
$S = (I \ \overline{II} \ \overline{IV} \ III)$	$\pi = (II \ \overline{I} \ \overline{IV} \ III)$

$$E = (III \ IV \ I \ II)$$

Elements of the unordered operations group.

TABLE 3.2

RIGHT

E
 Ø J T G B π I I N C₂ R V U S Y X W Z F D K T₂ L M O C₁ T₁ C Q P A₁ A₂ I
 π P Q B G Ø U W D M T₂ H K O L N T₁ C₁ C₂ S V X R Y F Z A₂ T J I C A₁
 B Q P π Ø G R Y F H K M T₂ N T₁ O L C₂ C₁ V S Z U W D X A₁ J T C I A₂
 D A A₁ F C₂ C₁ W U π N L O T₁ H T₂ M K Ø G X Z S Y R B V P C I J T Q
 F A A₂ D C₁ C₂ Y R B O T₁ N L M K H T₂ G Ø Z X V W U π S Q I C T J P
 G T J Ø π B M O C₁ U S R V W Z Y X D F T₂ K T₁ H N C L I P Q A₂ A₁ C
 H K T₂ M V S Ø C₂ N Q B P π A₁ D A₂ F Z X J T C G C₁ O I L R U Y W T₁
 J Ø G T Q P K L C V R S U Z W X Y A₁ A₂ H M N T₂ T₁ I O C₂ B π F D C₁
 K H M T₂ R U J C L B Q π P F A₂ D A₁ Y W Ø G C₂ T I T₁ C₁ N V S Z X O
 L N O T₁ W Y C J K D A₂ F A₁ π Q B P U R C₂ C₁ Ø I T T₂ G H X Z S V M
 M T₂ K H S V G C₁ O P π Q B A₂ F A₁ D X Z T J I Ø C₂ N C T₁ U R W Y L
 N L T₁ O X Z C₂ Ø H A₂ D A₁ F P B Q π S V C I J C₁ G M T K W Y U R T₂
 O T₁ L N Z X C₁ G M A₁ F A₂ D Q π P B V S I C T C₂ Ø H J T₂ Y W R U K
 P π B Q T J S X A₂ T₂ M K H T₁ N L O I C U R W V Z A₁ Y D G Ø C₁ C₂ F
 Q B π P J T V Z A₁ K H T₂ M L O T₁ N C I R U Y S X A₂ W F Ø G C₂ C₁ D
 R V S U K T₂ B F Y J Ø T G C C₁ I C₂ L T₁ Q P A₁ π D W A₂ Z H M N O X
 S U R V M H P A₂ X G T Ø J C₁ C C₂ I O N π B D Q A₁ Z F W T₂ K T₁ L Y
 U S V R T₂ K π D Y T G J Ø I C₂ C₁ C₁ T₁ L P Q A₂ B F Y A₁ X M H O N Z
 V R U S H M Q A₁ Z Ø J G T C₂ I C₁ C C₁ N O B π F P A₂ X D Y K T₂ L T₁ W
 W X Z Y L T₁ D π U C C₂ I C₁ J G T Ø K T₂ A₂ A₁ P F B R Q S N O H M V
 X W Y Z N O A₂ P S C₂ C₁ I Ø T G J H M D F π A₁ Q V B U L T₁ K T₂ R
 Y Z X W T₁ L F B R I C₁ C C₂ T Ø J G T₂ K A₁ A₂ Q D π U P V O N M H S
 Z Y W X O N A₁ Q V C₁ I C₂ C G J Ø T M H F D B A₂ P S π R T₁ L T₂ K U
 T G Ø J P Q T₂ T₁ I S U V R X Y Z W A₂ A₁ M H O K L C N C₁ π B D F C₂
 T₂ M H K U R T I T₁ π P B Q D A₁ F A₂ W Y G Ø C₁ J C L C₂ O S Y X Z N
 T₁ O N L Y W I T T₂ F A₁ D A₂ B P π Q R U C₁ C₂ G C J K Ø M Z X V S H
 A₂ D F A₁ C I X S P L N T₁ O K M T₂ H J T W Y U Z V Q R π C₂ C₁ Ø G B
 A₁ F D A₂ I C Z V Q T₁ O L N T₂ H K M T J Y W R X S P U B C₁ C₂ G Ø π
 C C₂ C₁ I A₂ A₁ L K J X W Z Y S R V U P Q N O H T₁ T₂ T M Ø D F π B G
 C₂ C I C₁ D F N H Ø W X Y Z U V R S π B L T₁ K O M G T₂ J A₂ A₁ P Q T
 C₁ I C C₂ F D O M G Y Z W X R S U V B π T₁ L T₂ N H Ø K T A₁ A₂ Q P J
 I C₁ C₂ C A₁ A₂ T₁ T₂ T Z Y X W V U S R Q P O V M L K J H G F D B π Ø E

TABLE 3.3

Operations Multiplication Table

(Side) (Top) = Operation

Another feature inherent in this formulation of the group, which is very useful in searching for new kernels, is the interlace breakdown into the group. This is useful because a kernel may be considered as the interlace of two half length codes where, as will be shown in Chapter 7, the number of zeros and ones in each half length code are easily predetermined.

Although this group of operations is indeed a group, it has never been proved formally. In order for a collection of elements to be a group, they must satisfy the following four conditions:¹⁵

1. A group has closure.
2. A group is associative under the group operation.
3. A group has an identity element.
4. Each element in a group has an inverse.

That this operation group has closure is demonstrated by Table 3.3 which is the full multiplication table of this group of order 32. The elements under the operation multiplication are associative since they are expressible as matrices and matrix multiplication is associative. The element I is the identity, and as seen from Table 3.3 all elements have an inverse. Therefore these 32 elements have satisfied all the requirements for a group.

There are three special subgroups of order 4 which within the subgroup deal with only one type of operation, $T_1 T_2 T I$, $A_1 A_2 C I$, $C_1 C_2 C I$. All of these subgroups have the unusual property that each element in the subgroup is its own inverse. This group of order 32 has been identified as isomorphic with Senior's group number 44.¹⁶ This will be discussed in some detail at the end of this chapter.

A study of complementary codes in (I II III IV) form, or as we will call it sequence quadruple form, reveals many interesting properties. Some of these properties and their formal proofs will follow.

In these proofs the complementary pair (A, B) will be assumed to be of length n. The symbol L_A will mean $\sum_{i=1}^{i=j} (a_i \oplus a_{n+i-j} \oplus 1)$ for all j, $1 \leq j \leq n-1$ and similarly $U_B = \sum_{i=1}^{i=j} b_i \oplus b_{n+i-j}$.

Theorem 3.1

If (A, B) are a complementary sequence pair and if either (I, II) or (III, IV) in sequence quadruple form are complementary, then the other pair must also be complementary.

1. Assume (I, II) are complementary.
2. $L_I = U_{II}$ by the definition of complementarity.
3. $U_I = L_{II}$ by the definition of complementarity.
4. Adding the equations in steps 2 and 3 gives $L_I + L_{II} = U_I + U_{II}$.
5. Since only even spacings are concerned in the sequence quadruple form, when related back to the original (A, B) pair, the likes of A are equal to the unlikes of A for all even spacings. This will be expressed as $L_A = U_A$ for even space.
6. $L_A = U_B$ and $U_A = L_B$ by the definition of complementarity.
7. Therefore $L_B = U_B$ for even spacings.
8. $L_{III} + U_{III} = k_j$ since the total number of matches at any spacing is equal to the likes plus the unlikes, and is dependent upon j.
9. $L_{IV} + U_{IV} = k_j$.
10. Adding 8 and 9 gives $L_{III} + U_{III} + L_{IV} + U_{IV} = (L_{III} + L_{IV}) + (U_{III} + U_{IV}) = 2k_j$.
11. $L_{III} + L_{IV} = k_j$, by step 7.
12. $L_{IV} - U_{III} = 0$ by subtracting the equation in step 8 from that in step 11.
13. Therefore $L_{IV} = U_{III}$ and the (III, IV) pair is complementary. Since the proof would have been the same if (III, IV) were chosen complementary rather than (I, II), the general statement is true.

* * *

Theorem 3.2

If (A,B) are a complementary sequence pair in sequence quadruple form and if one of the pair (I,II) or (III,IV) is not complementary neither is the other.

1. Assume (I,II) not complementary.
2. There are two possibilities for (III,IV) either complementary or not complementary.
3. Assume (III,IV) are complementary, then by the Theorem 3.1 the (I,II) pair must also be complementary, but this is contrary to assumption.
4. Therefore (III,IV) cannot be a complementary pair.

* * *

Theorem 3.3

A necessary and sufficient condition for (I,II) and (III,IV) to be two complementary pairs when a complementary sequence pair is written in the sequence quadruple form is if the likes for even spacing of one of the code pair is equal to the unlikes at the same spacing for the same code of the pair. First the sufficient portion:

1. $L_A = U_A$ for even spacings.
2. $L_I + L_{II} = L_A$ for even spacings.
3. $U_I + U_{II} = U_A = L_A$ for even spacings.
4. $L_I + L_{II} + U_I + U_{II} = 2L_A$ for even spacings by adding the equation in step 2 and step 3.
5. $L_I + U_I = k_j$ since the total of likes and unlikes must equal the possibilities.
6. $L_{II} + U_{II} = k_j$.
7. $L_I + U_I + L_{II} + U_{II} = 2k_j = 2L_A$ by adding the equations in steps 5 and 6 and comparing to those in step 4.
8. $L_I + L_{II} = k_j$ rewriting step 2 with k_j substituted for L_A .
9. $L_{II} - U_I = 0$ subtracting the equation in step 5 from that in step 8.

10. Therefore $L_{II} = U_I$ and (I, II) are a complementary pair and by Theorem 3.1 (III, IV) are also a complementary pair.
11. For the proof of the necessary condition assume that (I, II) are complementary and that $U_A < L_A$ for some even spacing.
12. Therefore $U_I + U_{II} < L_I + L_{II}$ for some even spacing.
13. However, by assumption (I, II) are complementary and $L_I = U_{II}$ and $U_I = L_{II}$.
14. Adding the two equations in step 13 gives $L_I + L_{II} = U_I + U_{II}$.
15. This is incompatible with step 12, therefore it is necessary that $U_A = L_A$ or $U_B = L_B$ for all even spacings in order to have (I, II) and (III, IV) complementary.

* * *

Theorem 3.4

If the sequence quadruple form of a complementary pair can be transformed into the pattern $(I \ II \ I \ \overline{II})$, either the original pair is the quad or (I, II) and (II, IV) are complementary pairs.

1. Given that $(I \ II \ III \ IV)$ is transformable into $(I \ II \ I \ \overline{II})$, and is so transformed.
2. $L_I + L_{II} + L_{III} + L_{IV} = U_I + U_{II} + U_{III} + U_{IV} = 2k_j$ by the supplementary property, Theorem 6.1.
3. $L_{II} = L_{\overline{II}}$ and $U_{II} = U_{\overline{II}}$ since $a_i \oplus a_{i+j} = \bar{a}_i \oplus \bar{a}_{i+j}$ by Theorem 2.1.
4. Therefore $2L_I + 2L_{II} = 2U_I + 2U_{II} = 2k_j$ by substituting into step 2, and using the condition of step 1.
5. $L_I + L_{II} = k_j$.
6. $L_I + U_I = k_j$ since the total of likes and unlikes must equal all possibilities.
7. $L_{II} - U_I = 0$ subtracting the equation in step 6 from that in step 5.
8. Therefore $L_{II} = U_I$ and (I, II) are complementary.
9. Since (I, II) are complementary, so are (I, \overline{II}) and any general transformations of (I, II) and (III, IV) .

* * *

This last property should be extremely useful in decomposing long codes, since it avoids the tedious task of checking to see if (I,II) are a complementary pair. Whenever a complementary pair expressed in sequence quadruple form has the pattern (I II I $\overline{\text{II}}$), certain additional transformations are allowed which are not true in general. These allowable operations are the time reversal of I and III, II and IV, or both. Also allowed as an operation is the altering of either even bits or odd bits for all four sequences. The proof of all these operations being allowable is quite simple, but does require an addition to the notation to cover these particular operations. Double or quadruple subscripts will be used to denote these operations, T_{13} , T_{24} and T_{1234} represent respectively the time reversal of (I,III), (II,IV) and the time reversal of both pairs. A_{14} represents the altering of the odd bits in all four sequences while A_{24} stands for the altering of even bits in all four sequences. Actually complementing any two, or all four of the four sequences will also leave a complementary pair when reassembled, but each of these operations is already accomplished by the general transformations.

Theorem 3.5

Whenever a complementary sequence pair is put into sequence quadruple and then transformed so that the resultant group pattern is (I II I $\overline{\text{II}}$), I and III, or II and IV, or both, can be time reversed and the sequence pairs formed on reassembly from the quadruple, after the operations (T_{13} T_{24} or T_{1234}), will still be complementary sequence pairs.

1. Given, (A,B) is a complementary pair of length n transformable in quadruple form to (I II I $\overline{\text{II}}$), and is so transformed.
2. (I,II) is a complementary sequence pair by Theorem 3.4.

3. Therefore $(\underline{I}, \underline{II})$, $(\underline{I}, \underline{\overline{II}})$ and $(\underline{\overline{I}}, \underline{II})$ are complementary pairs by Theorem 2.3.
4. From this it is seen that the original sequence quadruple after the operations T_{13}, T_{24} and T_{1234} are $(\underline{I} \ \underline{II} \ \underline{I} \ \underline{\overline{II}})$, $(\underline{\overline{I}} \ \underline{\overline{II}} \ \underline{I} \ \underline{\overline{II}})$ and $(\underline{I} \ \underline{\overline{II}} \ \underline{\overline{I}} \ \underline{II})$. When interlaced in the customary manner to form pairs, the pairs they form will be complementary sequence pairs by Theorem 2.8.

* * *

Theorem 3.6

Whenever a complementary sequence pair is put into sequence quadruple and then transformed so that the resultant group pattern is $(\underline{I} \ \underline{II} \ \underline{I} \ \underline{\overline{II}})$, the operation A_{14} , the altering of odd bits, or the operation A_{24} , the altering of even bits, will allow the quadruple upon reassembly to still be a complementary pair.

1. By hypothesis A and B are a complementary pair which when expressed in sequence quadruple form are transformable to $(\underline{I} \ \underline{II} \ \underline{I} \ \underline{\overline{II}})$, and is so transformed.
2. $(\underline{I}, \underline{II})$ is a complementary sequence pair by Theorem 3.4.
3. $(\underline{I}, \underline{II})$ when operated upon by either A_1 or A_2 is still a complementary pair by Theorem 2.4.
4. If the symbol $(1, 2)$ is used for the pair $(\underline{I}, \underline{II})$ altered it is seen that the pair formed from the interlace of $(1 \ 2 \ 1 \ \overline{2})$ is a complementary pair by Theorem 2.8.

* * *

These last two theorems should prove quite useful when trying to estimate an upper limit on the possible number of composite complementary sequences of any particular length. An example to illustrate the use of Theorem 3.5 might prove helpful. Taking pair number one from Appendix II,

$A=1101000111011110$
 $B=1000010010001011$
 $I=10001011=I$
 $II=01111011=II$
 $III=10001011=I$
 $IV=10000100=\overline{II}$

Since this is already in $(I \ II \ I \ \overline{II})$ form without transforming it,

T_{13} can be applied directly giving

$I=11010001$
 $II=01111011$
 $III=11010001$
 $IV=10000100.$

Reassembling gives

$A^1=1111001101010110$
 $B^1=1010011000000011$

which is identical with number three in Appendix II.

As an example, to illustrate the use of Theorem 3.6, applying A_{24} to the same (A,B) pair gives

$1.=11011110$
 $2.=00101110$
 $3.=11011110$
 $\overline{2}.=11010001,$

reassembling give

$A^2=1011011110111000$
 $B^2=1110001011101101.$

This is the same as number 6 in Appendix II with the transformation $TC_2=\emptyset$ applied.

Transform		Unordered Equivalent	Transform		Unordered Equivalent
1.	$I = (I \ II \ 1 \ \overline{II})$	A_2	17.	$R = (I \ \overline{II} \ \overline{II} \ \overline{I})$	O
2.	$A_1 = (\overline{I} \ II \ \overline{I} \ \overline{II})$	C	18.	$Q = (II \ \overline{I} \ \overline{II} \ \overline{I})$	J
3.	$A_2 = (I \ \overline{II} \ 1 \ II)$	I	19.	$P = (\overline{II} \ I \ II \ I)$	T
4.	$T_1 = (II \ 1 \ 1 \ \overline{II})$	S	20.	$O = (\overline{II} \ \overline{I} \ 1 \ \overline{II})$	R
5.	$T_2 = (I \ II \ \overline{II} \ I)$	X	21.	$N = (II \ 1 \ \overline{I} \ II)$	U
6.	$T = (II \ I \ \overline{II} \ 1)$	P	22.	$M = (\overline{I} \ \overline{II} \ \overline{II} \ I)$	Y
7.	$C_1 = (\overline{I} \ \overline{II} \ 1 \ \overline{II})$	F	23.	$L = (\overline{II} \ \overline{I} \ \overline{I} \ II)$	V
8.	$C_2 = (I \ II \ \overline{I} \ II)$	D	24.	$K = (\overline{I} \ \overline{II} \ II \ \overline{I})$	Z
9.	$C = (\overline{I} \ \overline{II} \ \overline{I} \ II)$	A_1	25.	$J = (\overline{II} \ \overline{I} \ II \ \overline{I})$	Q
10.	$Z = (II \ \overline{I} \ \overline{I} \ \overline{II})$	K	26.	$H = (I \ II \ II \ \overline{I})$	W
11.	$Y = (\overline{II} \ I \ \overline{I} \ \overline{II})$	M	27.	$G = (\overline{II} \ \overline{I} \ \overline{II} \ 1)$	B
12.	$X = (\overline{II} \ 1 \ I \ II)$	T_2	28.	$F = (I \ \overline{II} \ \overline{I} \ \overline{II})$	C_1
13.	$W = (II \ \overline{I} \ I \ II)$	H	29.	$D = (\overline{I} \ II \ 1 \ II)$	C_2
14.	$V = (\overline{I} \ II \ \overline{II} \ \overline{I})$	L	30.	$\pi = (II \ I \ II \ \overline{I})$	\emptyset
15.	$U = (\overline{I} \ II \ II \ 1)$	N	31.	$B = (\overline{II} \ I \ \overline{II} \ \overline{I})$	G
16.	$S = (I \ \overline{II} \ II \ I)$	T_1	32.	$\emptyset = (II \ \overline{I} \ II \ 1)$	π
				$E = (I \ \overline{II} \ I \ II)$	A_2, I

Redundance due to $(I \ II \ I \ \overline{II})$.

TABLE 3.4

The $(I \ II \ I \ \overline{II})$ form, or one of its transforms, appears quite often in the study of composite complementary sequences. Since two of the components are the same, and the other two are complements, it is reasonable to assume that some of the transforms are redundant, that is yield indistinct results. Table 3.4 lists all the transforms with the elements I, II and their complements. If (A,B) are taken as an ordered pair, the number of different codes formed from the transformation group are 32 instead of 64 because $A_2(I \ II \ I \ \overline{II})$ and $E(I \ II \ I \ \overline{II})$ are the same. If (A,B) are taken as an unordered pair the number of transforms forming different codes is decreased from 32 to 16. Table 3.4 lists each transform with its unordered mate.

The only possibility of having less than 16 unordered pairs of codes which are distinct would be if the group form could be transformed into the pattern $(I \ I \ I \ \overline{I})$. The only case known to the writer where this is true is the quad. For one version of the quad this is

$$\begin{aligned} A &= 11 & B &= 10 \\ I &= 1 \\ II &= 1 \\ III &= 1 \\ IV &= 0. \end{aligned}$$

Table 3.5 shows the eight ordered pairs, or if the cross index column is used the four unordered pairs, for the quad. The four possible code transformations for the quad are:

	1	2	3	4
A=	11	11	00	00
B=	10	01	10	01.

Pattern	Transforms with pattern for ordered pairs	Cross index number for unordered pairs
1. (I I I \bar{I})	I, T ₁ , H, π	3.
2. (\bar{I} I \bar{I} \bar{I})	A ₁ , Y, V, B	6.
3. (I \bar{I} I I)	A ₂ , W, S, ϕ , E	1.
4. (I I \bar{I} I)	T ₂ , T, C ₂ , N	8.
5. (\bar{I} \bar{I} I \bar{I})	C ₁ , O, K, J	7.
6. (\bar{I} \bar{I} \bar{I} I)	C, M, L, G	2.
7. (I \bar{I} \bar{I} \bar{I})	Z, R, Q, F	f.
8. (\bar{I} I I I)	X, U, P, D	4.

Redundance due to (I I I \bar{I}).

TABLE 3.5

This exhausts the transformations of special types and the redundancies caused by (I II I \bar{II}). The operations group will now be compared to Senior's group number 44 of length 32 as was mentioned earlier in this chapter.

The identification of any group with a group in Senior's list, is accomplished through establishing a one to one correspondence between group generators.¹⁶ As will be demonstrated, Senior's group number 44 of order 32 is in one to one correspondence with the unordered operations group of complementary sequences expressed in sequence quadruple form. Therefore the two groups are isomorphic.

Senior's generators will be given in lower case letters, while those of the operations group will be in their standard upper case symbols. Five generators are required for Senior's number 44; let these be a, b, c, d, e. The following relationships between generators are required for Senior's 44:

1. $a^2=b^2=c^2=d^2=e^2=1$ where 1 is the identity.
2. $b^{-1}ab=a, c^{-1}ac=a, d^{-1}ad=a$ where x^{-1} is the inverse of x.

3. $c^{-1}bc=b, d^{-1}bd=b,$
4. $d^{-1}cd=c,$
5. $e^{-1}ae=a, e^{-1}be=b,$
6. $e^{-1}ce=ca, e^{-1}de=bd.$

A check of the above requirements shows that a, b, c, d form an abelian group of order 16 while the "e" generator is commutative only with a, b and is non-commutative with c, d .

Let $a=C_1, b=C_2, c=T_1, d=T_2, e=A_1$. From Table 3.3 it is seen that the following relations are true.

1. $C_1^2 = C_2^2 = T_1^2 = T_2^2 = A_1^2 = I$
2. $C_2 C_1 C_2 = C_1, T_1 C_1 T_1 = C_1, T_2 C_1 T_2 = C_1.$
3. $T_1 C_2 T_1 = C_2.$
4. $T_2 T_1 T_2 = T_1.$
5. $A_1 C_1 A_1 = C_1, A_1 C_2 A_1 = C_2.$
6. $A_1 T_1 A_1 = T_1 C_1 = 0, A_1 T_2 A_1 = C_2 T_2 = H.$

This one to one correspondence between the generators of Senior's group number 44 and a set of generators from the unordered operations group on complementary sequences is necessary and sufficient to prove the two groups are isomorphic.

The diagrams for Senior's group number 44 of order 32 and those for its subgroups were used as a guide line to check the generation of subgroups from the operations group of unordered complementary sequence pairs. These subgroups of order 1, 2, 4, 8 and 16 are shown in Appendix IV. All groups that were shown in Senior's diagrams are included in these charts; however, Senior's diagrams included only normal subgroups and, although much care was taken in the generations of the subgroups of the operations group, there is a possibility that Appendix IV is not exhaustive due to some undetected non-normal subgroups.

Senior in conjunction with others has rewritten his work and changed his designating method.¹⁷ Under this new method his number 44 of length 32 is now $\Gamma_4 a_1$. The ordered operations group of length 64 is also identified as $\Gamma_{25} a_1$.

An identification of the group with Miller's number 23 of order 32 was also accomplished for those who prefer the substitution group method of identification.¹⁸

CHAPTER IV

HAMMING DISTANCES OF COMPLEMENTARY SEQUENCES

The Hamming distance is a well known property of certain classes of codes.¹⁰ This very useful property has had most of its applications in the fields of binary error detecting and error correcting codes. This chapter will demonstrate the usefulness of Hamming distances in the field of complementary sequences, as both the code pairs and their sequence quadruple formulation have certain invariances in Hamming distance under various conditions.

The Hamming distance, $D(U, V)$, of two binary vectors or sequences U and V is defined as the number of positions in which these two binary vectors differ.

$$D(U, V) = \sum_{i=1}^{i=n} u_i \oplus v_i \quad (4.1)$$

For example, the Hamming distance of the two binary vectors

$$U=1011001$$

$$V=0111001$$

$$\text{is } D(U, V) = \sum_{i=1}^7 u_i \oplus v_i = (1+1+0+0+0+0+0) = 2$$

The Hamming weight or just weight of a binary vector is its distance from the null vector.

Theorem 4.1

All complementary sequence pairs of length n have a Hamming distance of $n/2$.

By hypothesis

$$A=a_1a_2a_3\ldots\ldots a_n$$

and

$$B=b_1b_2b_3\ldots\ldots b_n$$

A and B form a complementary sequence pair.

1. All complementary sequence pairs satisfy the parity check

$$a_i \oplus a_{n-i+1} \oplus b_i \oplus b_{n-i+1} = 1, \text{ for all } i, 1 \leq i \leq n/2 \quad \text{by equation 2.4.}$$

2. Since mod 2 addition is commutative

$$a_i \oplus b_i \oplus a_{n-i+1} \oplus b_{n-i+1} = 1.$$

3. Therefore either $(a_i \oplus b_i) = 1$ or $(a_{n-i+1} \oplus b_{n-i+1}) = 1$, but not both.

4. For each $i, 1 \leq i \leq n/2$ one of the two vector positions in step 3 will have a one and the other vector position a zero. Summing up over all possible i , gives $n/2$ for the Hamming distance.

* * *

When a complementary sequence pair is decomposed into a sequence quadruple, each pair of sequences in the quadruple, which form one of the complementary pairs, has a Hamming distance which will be shown to be characteristic of the code length. The sum of the two pairs of Hamming distances from the quadruple is an invariant and is equal to $n/2$ the Hamming distance of the complementary sequence pair.

For example,

$$A = 1001010001$$

$$B = 1000000110$$

are a complementary sequence pair of length 10.

Decomposing them into the standard sequence quadruple, (I II III IV), gives

$$I = 10000$$

$$II = 10110$$

$$III = 10001$$

$$IV = 01000$$

$$\begin{aligned}
I \oplus II &= 00110 & D(I, II) &= 2 \\
III \oplus IV &= 11001 & D(III, IV) &= 3 \\
I \oplus II \oplus III \oplus IV &= 11111 & D(I \oplus II, III \oplus IV) &= 5.
\end{aligned}$$

This distance is half the length of the complementary sequence pair, and is required because of the necessary condition of the parity check.

If the distances (2, 3) for the pairs given in the above example are taken as an unordered rather than an ordered pair, this (2, 3) pair will be invariant through all general transformation operations.

All of the general transformations on the sequence quadruples have one or more of the following properties:

1. Change the order of (I, II) or (III, IV) or both, for example, T_1 , T_2 , T .
2. Complement the pairs (I, II) or (III, IV) or both, for example, C_1 , C_2 , C .
3. Complement one from the pair (I, II) the other from the pair (III, IV), for example, A_1 , A_2 .
4. Exchange (I, II) for (III, IV), for example, E .

Since all operations in the group can be generated by multiplication from the set listed above, an examination of the effect on Hamming distance by these transformations will be adequate to prove the invariance.

Theorem 4.2

A change in order of (I, II) or (III, IV) or both will not change the Hamming distances of the pairs (I, II) or (III, IV).

1. By definition $D(U, V) = \sum_{i=1}^n u_i \oplus v_i$
2. $u_i \oplus v_i = v_i \oplus u_i$ modulo 2 addition is commutative
3. $D(U, V) = \sum_{i=1}^n u_i \oplus v_i = \sum_{i=1}^n v_i \oplus u_i = D(V, U)$

*

*

*

Theorem 4.3

Complementing (I, II) or (III, IV) or both will not change the Hamming distances of the pairs (I, II) or (III, IV).

1. $u_i \oplus v_i = \bar{u}_i \oplus \bar{v}_i$ by Theorem 2.1
2. Therefore $\sum u_i \oplus v_i = \sum \bar{u}_i \oplus \bar{v}_i$
3. and $D(U, V) = D(\bar{U}, \bar{V})$

* * *

If it is assumed that the length of the sequence quadruples formed from the complementary sequence pairs have a length K and that the distance of the first pair is R , then the following theorem can be proved:

Theorem 4.4

Complementing one of the pair (I, II) and one of the pair (III, IV) results in the exchange of their Hamming distances, or expressed differently, gives the same distance pair in opposite order.

1. Let $D(I, II) = R$, therefore
 $D(III, IV) = K - R$
as the parity check requires $D(I \oplus II, III \oplus IV) = K$
2. Complement one of the first pair in the (I II III IV) form, and also complement one of the second pair.
Each vector position that was formerly alike and summed to zero is now different and sums to one.
Similarly each vector position that was formerly different and summed to one is now alike and sums to zero.
3. $D(\bar{I}, II) = D(I, \bar{II}) = K - R$ from 2.
4. $D(\bar{III}, IV) = D(III, \bar{IV}) = K - (K - R) = R$ from 2.

* * *

Theorem 4.5

The exchange operation has the effect of changing the order of Hamming distances of the pairs (I, II) and (III, IV) when expressed in sequence quadruple form.

1. The pairs (I, II) and (III, IV) are not changed in any way except by position in the exchange operation.
2. The only way to effect the Hamming distances of a pair of vectors is to change some of the components of the vectors.
3. Therefore the two weights are the same but reversed in order.

* * *

An examination of the Hamming distances for the known kernels of 2, 10 and 26 is now in order as these distances are a characteristic of the kernel, as are the vectors formed in the process of obtaining the distances by the modulo 2 sums.

The first case to examine is the trivial case for the kernel of length 2.

A=11 B=10

Broken down into sequence quadruple representation

I=1 D(I, II)=0
 II=1
 III=1 D(III, IV)=1
 IV=0

There are two kernels for length 10 to be examined.

A=1001010001
 B=1000000110

I=10000 I ⊕ II=00110 D(I, II)= 2
 II=10110
 III=10001 III ⊕ IV=11001 D(III, IV)= 3
 IV=01000

and

A=0101000011
 B=0000100110
 I=00001 I ⊕ II=10010 D(I, II)= 2
 II=10011
 III=00101 III ⊕ IV=01101 D(III, IV)= 3

Both kernels of length 10 have the unordered Hamming distances of (2,3) when expressed in sequence quadruple representation. However, their "Hamming vectors", the unordered pairs (00110,11001) and (10010,01101) are different and are characteristic of the kernel of their origin under all possible transformation operations. Hamming vectors will be considered at length in the next chapter and proof of the above statement will be deferred until then.

There is only one kernel of length 26 (Chapter 7) and it is

A=01001101111010111100111010

B=10110010000111111100111010

I=0010111110111

I \oplus II=0000001101010

D(I, II) = 4

II=0010110011101

III=1101001110111

III \oplus IV=1111110010101

D(III, IV)= 9

IV=0010111100010

This is a complete listing of the Hamming distances for all known kernels and it is exhaustive for $n=2,10,18$ and 26. It is worthwhile to note that the sum of the unordered pairs of distances is always equal to one half the length of the code and that the Hamming vectors of the pairs are complements. Both of these conditions are due to the parity check being a necessary condition.

A natural extension of the Hamming distances of kernels is the consideration of Hamming distances of composite codes when put into sequence quadruple form. Six different code pairs of length eight are listed in Table 4.1. These pairs will be examined for a possible invariance. After this invariance is noted a general proof will be given for the characteristic distance of composite codes.

The first four examples in the table when decomposed into sequence quadruple representation have (I,II) and (III,IV) as code pairs which are still complementary. These codes are half the length of the original pair, and being complementary, have a Hamming distance one half of their length, or $n/4$ as compared to the original codes of length eight.

(a)	A=11111001	H=(00110011)	
	B=11001010		
	I=1110	I \oplus II=0101	D(I, II) = 2
	II=1011		
(b)	III=1011	III \oplus IV=1010	D(III, IV) = 2
	IV=0001		
	A=11110110	H=(00110011)	
	B=11000101		
(c)	I=1101	I \oplus II=1010	D(I, II) = 2
	II=0111		
	III=1000	III \oplus IV=0101	D(III, IV) = .2
	IV=1101		
(d)	A=11101101	H=(00001111)	
	B=11100010		
	I=1110	I \oplus II = 0011	D(I, II) = 2
	II=1101		
(e)	III=1101	III \oplus IV = 1100	D(III, IV) = 2
	IV=0001		
	A=11011110	H=(00001111)	
	B=11010001		
(f)	I=1011	I \oplus II = 1100	D(I, II) = 2
	II=0111		
	III=1000	III \oplus IV = 0011	D(III, IV) = 2
	IV=1011		
(g)	A=11101011	H=(00001111)	
	B=11100100		
	I=1111	I \oplus II = 0110	D(I, II) = 2
	II=1011		
(h)	III=1100	III \oplus IV = 1001	D(III, IV) = 2
	IV=0101		
(i)	A=10111110	H=(00001111)	
	B=10110001		
	I=1111	I \oplus II=1001	D(I, II) = 2
	II=0110		
(j)	III=1100	III \oplus IV=0110	D(III, IV) = 2
	IV=1010		

Some Hamming distances and vectors
of codes of length 8.

TABLE 4.1

An examination of the last two examples in the table shows that although the (I, II) and (III, IV) pairs are not complementary their Hamming distance is still $n/4$ when compared to the original codes of length eight. That this feature is true in general will now be proven by examining and exhausting all the known ways of generating composite codes.

Theorem 4.6

Every composite code of length n (generated by one of the standard methods) when expressed in the sequence quadruple representation will have $D(I, II) = D(III, IV) = n/4$.

There are presently known four general ways of generating composite codes from shorter codes, (there is also the method applicable only to codes of length 2^r given at the end of the proof). These are:

1. Time Sequence

$$S_1 = AB$$

$$S_2 = A\bar{B}$$

2. Interlace

$$T_1 = a_1 b_1 a_2 b_2 \dots a_n b_n$$

$$T_2 = a_1 \bar{b}_1 a_2 \bar{b}_2 \dots a_n \bar{b}_n$$

3. Time Sequence Exponential

$$U_1 = A^{c_1} A^{c_2} \dots A^{c_m} B^{d_1} B^{d_2} \dots B^{d_m}$$

$$U_2 = A^{d_m} A^{d_{m-1}} \dots A^{d_1} B^{c_m} B^{c_{m-1}} \dots B^{c_1}$$

4. Interlace Exponential

$$V_1 = A^{c_1} B^{d_1} A^{c_2} B^{d_2} \dots A^{c_m} B^{d_m}$$

$$V_2 = A^{d_m} B^{c_m} A^{d_{m-1}} B^{c_{m-1}} \dots A^{d_1} B^{c_1}$$

Throughout this proof it will be considered that (A, B) are a complementary pair of length r and (C, D) are a complementary pair of length m .

The parity check $a_i \oplus a_{n+1-i} \oplus b_i \oplus b_{n+1-i} = 1$ is a necessary condition for a sequence pair to be complementary. One of the simplest methods for making the parity check is to fold both codes in the middle and double them back on themselves.⁹ This procedure is shown in Figure 4.1. Each

$$\begin{array}{ccccccc}
 a_1 & a_2 & a_3 & \dots & a_i & \dots & a_{n/2-1} & a_{n/2} \\
 a_n & a_{n-1} & \dots & a_{n+1-i} & \dots & a_{n/2+1} & a_{n/2+1} \\
 b_1 & b_2 & b_3 & \dots & b_i & \dots & b_{n/2-1} & b_{n/2} \\
 b_n & b_{n-1} & \dots & b_{n+1-i} & \dots & b_{n/2+1} & b_{n/2+1}
 \end{array}$$

Foldover Parity Check Method

Figure 4.1

column in Figure 4.1 satisfies the parity check. If code pairs are written in the sequence quadruple form such as in Figure 4.2, it is seen that the columns of this array are the same as the column in the first array with rearrangements in order provided it is allowed that the columns

$$\begin{array}{ccc}
 a_r & & a_u \\
 a_u & \text{and} & a_r \\
 b_r & & b_u \\
 b_u & & b_r
 \end{array}$$

are the same. Since modulo 2 addition is the operation used in making the parity check and in determining Hamming distance, and the use of each column involves only modulo 2 addition which is commutative, the two columns can be considered the same. Therefore the fold-over method of grouping and the sequence quadruple form of grouping have invariant columns, order being ignored.

$$\begin{array}{ccccccc}
a_1 & a_3 & a_5 & \dots & a_{n-3} & a_{n-1} & \\
a_n & a_{n-2} & \dots & a_4 & a_2 & & \\
b_1 & b_3 & \dots & b_{n-3} & b_{n-1} & & \\
b_n & b_{n-2} & \dots & b_4 & b_2 & &
\end{array}$$

Sequence Quadruple Form
of Parity Check

FIGURE 4.2

Now consider the composite generating methods.

1. Time Sequence

$$S_1 = AB$$

$$S_2 = A\bar{B} \quad \text{writing in the fold-over parity check form}$$

$$\begin{array}{ccccccc}
a_1 & a_2 & a_3 & \dots & a_{r-2} & a_{r-1} & a_r \\
b_r & b_{r-1} & \dots & b_3 & b_2 & b_1 &
\end{array} \quad D(A, B) = r/2$$

$$\begin{array}{ccccccc}
a_1 & a_2 & a_3 & \dots & a_{r-2} & a_{r-1} & a_r \\
\bar{b}_r & \bar{b}_{r-1} & \dots & \bar{b}_3 & \bar{b}_2 & \bar{b}_1 &
\end{array} \quad D(A, \bar{B}) = r/2$$

The top pair are complementary and therefore have a distance of $r/2$, similarly the bottom two rows are complementary and have a distance of $r/2$ by Theorem 4.1. To write the time sequence code in sequence quadruple form would not change the distance since it would be purely a reordering of the parity check columns. Therefore $D(I, II)$ would equal $r/2$ as would $D(III, IV)$. In terms of the composite code of length n $D(I, II) = D(III, IV) = r/2 = n/4$ whether or not (I, II) and (III, IV) happen to form complementary pairs.

2. For the interlace scheme the (I, II) pair and the (III, IV) pair are obviously complementary sequences by construction and will have a distance of $r/2 = n/4$.

3. Writing the time sequence exponential pair in fold-over form

gives
$$\begin{array}{ccccccc}
U_1 = & A^{c_1} & A^{c_2} & \dots & A^{c_m} & & \\
& \underline{B}^{d_m} & \underline{B}^{d_{m-1}} & \dots & \underline{B}^{d_1} & &
\end{array}$$
where the symbol \underline{B} means B time reversed.

$$U_2 = \begin{matrix} d_m & d_{m-1} & & d_1 \\ A & A & \dots & A \\ \bar{c}_1 & \bar{c}_2 & & \bar{c}_m \\ \underline{B} & \underline{B} & \dots & \underline{B} \end{matrix}$$

The B's are all time reversed but the time reversal of one of the complementary pairs does not change the complementary property $D(A, B) = D(A, \underline{B}) = r/2$. For each bit of the C code and D code there is a distance $r/2$ since every A is matched with some transformation of B for each of these bits. There are a total of m bits in the C code and in the D code, therefore the total distance of U_1 folded back on itself is $m \times r/2 = mr/2$. Similar reasoning gives U_2 folded back on itself a distance of $mr/2$. A rearrangement of the columns will not change the Hamming distances. Therefore $D(I, II) = mr/2$ and $D(III, IV) = mr/2$, since the total length of the code is $n = 2mr$, $D(I, II) = D(III, IV) = n/4$.

The interlace exponential scheme is the last general method of code generation to be considered. Written in the fold-over form it gives

$$V_1 = \begin{matrix} c_1 & d_1 & c_2 & d_2 & \dots & c_{m/2} & d_{m/2} \\ A & B & A & B & \dots & A & B \\ \underline{B} & \underline{A} & \underline{B} & \underline{A} & \dots & \underline{B} & \underline{A} \\ d_m & c_m & d_{m-1} & c_{m-1} & \dots & d_{m/2+1} & c_{m/2+1} \end{matrix}$$

$$V_2 = \begin{matrix} d_m & \bar{c}_m & d_{m-1} & \bar{c}_{m-1} & \dots & d_{m/2+1} & \bar{c}_{m/2+1} \\ A & B & A & B & \dots & A & B \\ \underline{B} & \underline{A} & \underline{B} & \underline{A} & \dots & \underline{B} & \underline{A} \\ \bar{c}_1 & d_1 & \bar{c}_2 & d_2 & \dots & \bar{c}_{m/2} & d_{m/2} \end{matrix}$$

Again each A is matched with some transform of B, $D(A, B) = D(A, \underline{B}) = D(A, B)$, etc. $= r/2$. As before, rearranging columns into sequence quadruple form will not effect the Hamming distances. The distance of the top two rows is $mr/2$ and the bottom two rows (V_2) also is equal to $mr/2$. The total length of the code is $2mr$, therefore $D(I, II) = D(III, IV) = mr/2 = n/4$.

5. When composite codes are of length 2^r , they are sometimes formed in a special way from the interlaced sections of a complementary

pair. These sections or pieces of the codes must be of length $p=2^j$ where $j < r$. Using the symbols $A_p = a_1 a_2 a_3 a_4 \dots a_p$, $A_{2p} = a_{p+1} a_{p+2} \dots a_{2p}$ and so, the following pair would be complementary if A and B formed a complementary pair.

$$S_{1p} = A_p B_p A_{2p} B_{2p} \dots A_{np/p} B_{np/p}$$

$$S_{2p} = A_p \bar{B}_p A_{2p} \bar{B}_{2p} \dots A_{np/p} \bar{B}_{np/p}$$

Rather than try to use the general term to prove that $D(I, II) = D(III, IV) = n/4$ as was done in the previous cases, a short general example of length 8 will be used. These two codes (A, B) of length eight will be interlaced four bits at a time ($p=2$) to form a code of length 16.

$$S_{14} = a_1 a_2 a_3 a_4 b_1 b_2 b_3 b_4 a_5 a_6 a_7 a_8 b_5 b_6 b_7 b_8$$

$$S_{24} = a_1 a_2 a_3 a_4 \bar{b}_1 \bar{b}_2 \bar{b}_3 \bar{b}_4 a_5 a_6 a_7 a_8 \bar{b}_5 \bar{b}_6 \bar{b}_7 \bar{b}_8$$

Writing this composite pair in fold-over form yields

$$S_{14} = a_1 a_2 a_3 a_4 b_1 b_2 b_3 b_4 \\ b_8 b_7 b_6 b_5 a_8 a_7 a_6 a_5$$

$$S_{24} = a_1 a_2 a_3 a_4 \bar{b}_1 \bar{b}_2 \bar{b}_3 \bar{b}_4 \\ \bar{b}_8 \bar{b}_7 \bar{b}_6 \bar{b}_5 a_8 a_7 a_6 a_5$$

$D(A, B) = 4$ by Theorem 4.1. S_{14} folded over is just a column rearrangement of the complementary pair (A, B), and since a rearrangement of columns does not effect Hamming distance, the Hamming distance of S_{14} folded-over is also 4. Similar reasoning holds for S_{24} .

Since the standard sequence quadruple form is again just a re-ordering of the columns, the distance of the (I, II) pair and the (III, IV) pair is $n/4$. For this example,

$$I = a_1 a_3 b_1 b_3 a_5 a_7 b_5 b_7 \quad D(I, II) = 4$$

$$II = b_8 b_6 a_8 a_6 b_4 b_2 a_4 a_2$$

$$III = a_1 a_3 \bar{b}_1 \bar{b}_3 a_5 a_7 \bar{b}_5 \bar{b}_7 \quad D(III, IV) = 4$$

$$IV = \bar{b}_8 \bar{b}_6 a_8 a_6 \bar{b}_4 \bar{b}_2 a_4 a_2 \quad D(I \oplus II, III \oplus IV) = 8.$$

Therefore any composite complementary sequence pair when written in sequence quadruple form has $D(I, II) = D(III, IV) = n/4$.

* * *

Although there is no proof that the 4 standard generating methods, plus the special one applicable only to 2^r codes, are the only way to form complementary codes which are composite, an exhaustive search of code lengths 16 and 20 revealed no codes which were not formed by these standard generating methods.

The full usefulness of the Hamming distance property is, of course, not yet known. The property of the distance of the complementary pair being equal to half the code length as shown in Theorem 4.1, is used in many of the proofs in the chapter on Hamming vectors. This property, $D(A, B) = n/2$, indicates that the code pairs are orthogonal. Because of this orthogonality property the leakage of an A code carrier into the B code filter, or vice versa, will have no effect on the output at exact match.

The primary use the writer has found for the Hamming distance of the (I, II) and (III, IV) pairs, has been the error check that this invariant property offered in decomposing codes by hand. One possible use for the invariance of $D(I, II)$ and $D(III, IV)$ could be in the search for kernels of new lengths. If a characteristic Hamming distance for unknown kernels could in some way be deduced, based only on length, this invariant property of distance would act as a most powerful screen in the search. The (I, II) and (III, IV) pairs are themselves orthogonal, unless the code is a kernel. This may provide other possible methods for searching for new codes. It is difficult to say in advance what utilization will be made of any invariant property, but it is fairly safe to say that the property will find use.

CHAPTER V

THE HAMMING VECTORS OF COMPLEMENTARY SEQUENCES

The Hamming distance between the code pairs in a complementary sequence was shown by Theorem 4.1 to be one half the length of the code. This Hamming distance was obtained by first finding what I shall call the Hamming vector, $H(A,B) = A \oplus B$ or for convenience just $H = A \oplus B$, and then counting the number of ones H contains. The Hamming vector is itself useful in recognizing the composition of a composite code when it is desired to break the code down into shorter lengths. This decomposition might be used to find previously unknown kernels, or to set an upper bound on the possible number of pairs of codes of any length.

An examination of each of the methods for generating longer codes from shorter ones as explained in Chapter 2 will show that each method has a characteristic Hamming vector form, although it may be necessary to time reverse one of the codes to obtain this form. The method of generating codes of length 2^n , which does not apply to complementary codes in general, will also be studied with its characteristic Hamming vector.

In the following proofs (A,B) form a complementary code pair of length n while (C,D) form a complementary code pair of length m . The symbol $(0)^n$ or just 0^n will mean a string of n zeros, similarly $(01)^n$ will symbolize a string of n zero ones, and $(a_i \oplus b_i)^n$ will be a string of n elements equal to $(a_i \oplus b_i)$.

Theorem 5.1

The time sequence form of generating a composite code has the characteristic Hamming vector $H = (0)^n (1)^n$.

Use the time sequence generating method on (A,B) to form

$$S_1 = AB$$

$$S_2 = A\bar{B}.$$

Take the Hamming vector of (S_1, S_2) , $H=S_1 \oplus S_2= 0^n 1^n$, this is
(5.1)
seen to be true by inspection and definition of modulo 2 sum.

* * *

Theorem 5.2

The interlace form of generating a composite code has the characteristic Hamming vector $H=(01)^n$.

Use the interlace generating method on (A,B) to form

$$T_1=a_1 b_1 a_2 b_2 a_3 b_3 \dots a_n b_n$$

$$T_2=a_1 \bar{b}_1 a_2 \bar{b}_2 a_3 \bar{b}_3 \dots a_n \bar{b}_n .$$

Take the Hamming vector of (T_1, T_2) , $H=T_1 \oplus T_2=(01)^n$, (5.2)
this is seen to be true by inspection and definition of modulo 2 sum.

* * *

Theorem 5.3

The time sequence exponential form of generating a composite code has the characteristic Hamming vector $H=(c_1 \oplus d_m)^n (c_2 \oplus d_{m-1})^n \dots (c_m \oplus d_1)^n (d_1 \oplus \bar{c}_m)^n \dots d_m \oplus \bar{c}_1)^n$.

Use the time sequence exponential method on (A,B) and (C,D) to form

$$U_1=A^{c_1} A^{c_2} \dots A^{c_m} B^{d_1} B^{d_2} \dots B^{d_m}$$

$$U_2=A^{d_m} A^{d_{m-1}} \dots A^{d_1} B^{\bar{c}_m} B^{\bar{c}_{m-1}} \dots B^{\bar{c}_1} .$$

Take the Hamming vector of (U_1, U_2) ,

$$H=U_1 \oplus U_2=(c_1 \oplus d_m)^n (c_2 \oplus d_{m-1})^n \dots (c_m \oplus d_1)^n (d_1 \oplus \bar{c}_m)^n \dots (d_m \oplus \bar{c}_1)^n .$$
(5.3)

This is seen to be true by inspection and definition of modulo 2 sum.

* * *

It is to be noted that each of the Theorems 5.1, 5.2, 5.3 have Hamming vectors which are anti-symmetric about their center. This is required to satisfy the fold-over method of making the parity check.

Theorem 5.3 also shows that along with the anti-symmetric characteristic the Hamming vector for time sequence exponential composition has unit clusters of n zeros or n ones and is also anti-symmetric in both halves. This is easily seen because the first sum is $(c_1 \oplus d_m)$ while the last sum to operate on the A code is $(c_m \oplus d_1)$, but it is known through the parity check that $c_1 \oplus d_m \oplus c_m \oplus d_1 = 1$. Therefore one of the sums $(c_1 \oplus d_m)$ or $(c_m \oplus d_1)$ must be zero and the other must be one. To summarize, for the time sequence exponential form of composition the first half of the Hamming vector is anti-symmetric about its center (the quarter length point) and the first half and the last half of the vector are also anti-symmetric.

Theorem 5.4 will concern the interlace exponential method of forming composite codes and it will be seen that this method will again form a Hamming vector anti-symmetric about its center, but with a different arrangement of the clusters of zeros and ones.

Theorem 5.4

The interlace exponential method of generating a composite code has the characteristic Hamming vector $H = (c_1 \oplus d_m)^n (d_1 \oplus \bar{c}_m)^n (c_2 \oplus d_{m-1})^n \dots (c_m \oplus d_1)^n (d_m \oplus \bar{c}_1)^n$.

Use the interlace exponential method on (A, B) and (C, D) to form

$$\begin{aligned} V_1 &= A^{c_1} B^{d_1} A^{c_2} B^{d_2} \dots A^{c_m} B^{d_m} \\ V_2 &= A^{d_m} B^{\bar{c}_m} A^{d_{m-1}} \dots A^{d_1} B^{\bar{c}_1} \end{aligned}$$

Take the Hamming vector of (V_1, V_2)

$$H = V_1 \oplus V_2 = (c_1 \oplus d_m)^n (d_1 \oplus \bar{c}_m)^n (c_2 \oplus d_{m-1})^n \dots (c_m \oplus d_1)^n (d_m \oplus \bar{c}_1)^n \quad (5.4)$$

This is seen to be true by inspection and definition of modulo 2 sum.

* * *

The interlace exponential method again gives a Hamming vector which is anti-symmetric about its center due to the condition of satisfying

the fold-over form of the parity check, but its clusters of zeros or ones are now $2n$ in length rather than just n as was the case for the time sequence exponential method of generation. This characteristic of clusters of $2n$ in length is caused by the necessary condition of the parity check for the (C,D) code pair. For example, the first two clusters are $(c_1 \oplus d_m)^n$ and $(d_1 \oplus \bar{c}_m)^n$ and the second two clusters are $(c_2 \oplus d_{m-1})^n$ and $(d_2 \oplus \bar{c}_{m-1})^n$. Applying the parity check to the $i=1$ and $i=2$ bits in the (C,D) pair gives the following two equations:

$$c_1 \oplus c_m \oplus d_1 \oplus d_m = 1 \quad \text{and} \quad c_2 \oplus c_{m-1} \oplus d_2 \oplus d_{m-1} = 1.$$

Adding one modulo 2 to both sides of both equations and rearranging gives $(c_1 \oplus d_m) \oplus (d_1 \oplus c_m \oplus 1) = 0$ and $(c_2 \oplus d_{m-1}) \oplus (d_2 \oplus c_{m-1} \oplus 1) = 0$. It is true in general that $x \oplus 1 = \bar{x}$; therefore $(c_1 \oplus d_m) \oplus (d_1 \oplus \bar{c}_m) = 0$ and $(c_2 \oplus d_{m-1}) \oplus (d_2 \oplus \bar{c}_{m-1}) = 0$. The sums $(c_1 \oplus d_m)$ and $(d_1 \oplus \bar{c}_m)$ must be alike to satisfy this constraint with each of the sums giving n ones if unlike, or n zeros if alike, and since the pairs are in time sequence the clusters must be in groups of $2n$. A similar reasoning will show that the parity check for $i=2$ gives identical results for the third and fourth terms in the Hamming vector for the interlaced exponential form of generating composite codes. Therefore equation 5.4 can be written

$$H(V_1, V_2) = (c_1 \oplus d_m)^{2n} (c_2 \oplus d_{m-1})^{2n} \dots (d_m \oplus \bar{c}_1)^{2n}. \quad (5.5)$$

Before going into the special case applicable only to codes of length 2^n it might prove helpful to give an example of generating a code by one method and then decomposing it, to show that it could have been composed by a second method. The time sequences exponential scheme will be the generating method used.

$$A=10$$

$$C=1001010001$$

$$B=11$$

$$D=1000000110$$

$$U_1=100101100110010101101100000000000111100$$

$$U_2=0110100101010101011000111111001100111100$$

$$H=U_1 \oplus U_2 = 1111111100110000000011111111001100000000$$

In passing it should be noted that the shortest clusters of zeros and ones in H are of length 2 and that H is anti-symmetric about its center.

If U_1 is now time reversed, indicated by $\underline{U_1}$, the complementary pair is now:

$$U_1 = 0011110000000000001101101010011001101001$$

$$\underline{U_1} = 01101001010101010101000111111001100111100$$

$$H = U_1 \oplus U_2 = 01 = (01)^{20}$$

This form of the Hamming vector indicates that H is composed of an interlace pair of length 20. Decomposing both $\underline{U_1}$ and U_2 by assuming an interlace pair gives

$$U_1 = 01100000010111010110$$

$$\underline{U_1} = 01100000011000101001$$

$$H = 00000000001111111111 = 0^{10}1^{10}$$

$$U_2 = 01100000010111010110$$

$$10011111100111010110$$

$$H = 11111111110000000000 = 1^{10}0^{10}$$

The Hamming vector of $\underline{U_1}$ indicates that this code of length 20 was composed by the time sequence method from codes of length 10. If the form AB is assumed for the codes of length 10 then

$$A = 0110000001$$

$$B = 0111010110$$

This (A,B) pair is seen to be the transformation $ETC_1 = EG$ on the original kernel of length 10 which was used to generate the code by a different method. Similarly the pair from U_2 is also a transformation of the original (C,D) pair. Therefore the given code of length 40 could have been generated by two different methods.

A second example will now be given using the same (A,B) and (C,D) kernels as were used in the first example, but this time utilizing the exponential interlace scheme of generation rather than the time sequence exponential form.

A=10

C=1001010001

B=11

D=1000000110

$V_1=1011010001001000010010000100011101111000$

$V_2=0100101110110111010001110100011101111000$

$H=V_1 \oplus V_2 = 1111111111111111000011110000000000000000$

Note that the smallest cluster is four, as would be expected of an interlace exponential scheme with the (A,B) pair of length two. The H vector is again anti-symmetric about its center. Time reversing V_1 gives

$V_1 = 0001111011100010000100100001001000101101$

$V_2 = 0100101110110111010001110100011101111000$

$H=V_{1t} \oplus V_2 = 01 = (01)^{20}$

As before this form of H indicates an interlace code pair. Decomposing both V_1 and V_2 into interlace code pairs gives

$V_1 = 00111101000100010110$

$\quad 01101000010001000011$

$H = 01010101010101010101 = (01)^{10}$

$V_2 = 00111101000100010110$

$\quad 10110111101110111100$

$H = 10101010101010101010 = (10)^{10}$

V_1 and V_2 both break down into interlaced pairs of length 10. The top row of V_1 breaks into A=0110000001, B=0111010110 which is the transformation EG operating on the original (C,D) pair as in the previous example. Each of the codes for V_2 would break down in similar fashion to some transformation of the original (C,D) code pair.

One use for the method just shown could be to search for larger kernels than those currently known. An example might be kernels of length fifty found through the use of time sequence exponential or interlace exponential, where both (A,B) and (C,D) are codes of length 10. These would form codes of length 200. From this as a starting point, a decomposition to codes of length 100 and then to length 50 might

be possible through this technique. This technique will be discussed much more thoroughly towards the end of this chapter, but it is necessary to consider the special case of 2^n code composition and the Hamming vectors of the sequence quadruple form before continuing with this interesting possibility.

The special generating method which is applicable only to codes of length 2^r is actually a generalization of both the time sequence and the interlace methods. As mentioned in the previous chapter, this method takes a code pair of length 2^r and from them forms a code pair of length 2^{r+1} . The allowable combinations are formed by interlacing pieces of the A code and the B code of length 2^m , where $m=0,1,2,\dots,r$.

As an example a code of length 8 can be used to generate codes of length 16. In this case $r=3$ and m has the possible values 0,1,2,3. These values of m give section lengths of 1,2,4,8 respectively.

$$A=a_1a_2a_3a_4a_5a_6a_7a_8$$

$$B=b_1b_2b_3b_4b_5b_6b_7b_8$$

for $m=0$,

$$S_{11}=a_1b_1a_2b_2a_3b_3a_4b_4a_5b_5a_6b_6a_7b_7a_8b_8$$

$$S_{21}=a_1\bar{b}_1a_2\bar{b}_2a_3\bar{b}_3a_4\bar{b}_4a_5\bar{b}_5a_6\bar{b}_6a_7\bar{b}_7a_8\bar{b}_8$$

$$H=S_{11} \oplus S_{21} = 0101010101010101 = (01)^8 = (0^1 1^1)^8$$

This is exactly the same as the interlace procedure which is a standard generating method.

The section lengths are 2 bits long for $m=1$.

$$S_{12}=a_1a_2b_1b_2a_3a_4b_3b_4a_5a_6b_5b_6a_7a_8b_7b_8$$

$$S_{22}=a_1a_2\bar{b}_1\bar{b}_2a_3a_4\bar{b}_3\bar{b}_4a_5a_6\bar{b}_5\bar{b}_6a_7a_8\bar{b}_7\bar{b}_8$$

$$H=S_{12} \oplus S_{22} = 0011001100110011 = (0^2 1^2)^4,$$

$m=2$ gives

$$S_{14} = a_1 a_2 a_3 a_4 b_1 b_2 b_3 b_4 a_5 a_6 a_7 a_8 b_5 b_6 b_7 b_8$$

$$S_{24} = a_1 a_2 a_3 a_4 \bar{b}_1 \bar{b}_2 \bar{b}_3 \bar{b}_4 a_5 a_6 a_7 a_8 \bar{b}_5 \bar{b}_6 \bar{b}_7 \bar{b}_8$$

$$H = S_{14} \oplus S_{24} = 0000111100001111 = (0^4 1^4)^2,$$

For $m=3$ the lengths are eight, which is the entire code.

$$S_{18} = AB$$

$$S_{28} = A\bar{B},$$

This is the same as the time sequence form giving a Hamming vector

$$H = 0000000011111111 = (0^8 1^8).$$

The Hamming vectors from this special construction are altering clusters of zeros and ones, each cluster being the length of the segment used in the generation.

Theorem 5.5

Complementary sequences of length 2^{r+1} formed from an (A, B) pair of length $n=2^r$ using the special generating method with segments of length m have a characteristic Hamming vector $H = (0^{m_1 m})^{2n/m}$.

Use the special generating method utilizing segments of length m of the pair (A, B) to form

$$S_{1m} = a_1 a_2 a_3 \dots a_m b_1 b_2 \dots b_m a_{m+1} a_{m+2} \dots a_{2m} b_{m+1} b_{m+2} \dots b_n$$

$$S_{2m} = a_1 a_2 a_3 \dots a_m \bar{b}_1 \bar{b}_2 \dots \bar{b}_m a_{m+1} a_{m+2} \dots a_{2m} \bar{b}_{m+1} \bar{b}_{m+2} \dots \bar{b}_n.$$

Take the Hamming vector of (S_{1m}, S_{2m})

$$H = S_{1m} \oplus S_{2m} = (0^{m_1 m})^{2n/m}.$$

This is seen to be true by inspection and definition of modulo 2 addition.

* * *

It now seems suitable for completeness to include a theorem which has been alluded to many times but not proven. This is the anti-symmetric property of all Hamming vectors of complementary sequences.

Theorem 5.6

The Hamming vector of a complementary sequence pair has a pattern which is anti-symmetric about the center of the Hamming vector.

1. $A = a_1 a_2 a_3 a_4 \dots a_{n-1} a_n$
 $B = b_1 b_2 b_3 b_4 \dots b_{n-1} b_n$ is a complementary pair of length n
2. $H(A, B) = h_1 h_2 h_3 h_4 \dots h_{n/2} \dots h_n$ by definition of Hamming vector
3. $a_i \oplus a_{n+1-i} \oplus b_i \oplus b_{n+1-i} = 1$ by equation 2.4
4. $a_i \oplus b_i = h_i$ and both by definition of h_i
5. $a_{n+1-i} \oplus b_{n+1-i} = h_{n+1-i}$
6. $h_i \oplus h_{n+1-i} = 1$
7. Add $(h_{n+1-i} \oplus 1)$ to both sides of the equation
8. $h_{n+1-i} = \bar{h}_i$

Make a change of reference to the center of the vector, which lies between $m/2$ and $m/2 + 1$, rather than the end by having $i = n/2 - r$ then
 $n+1-i = n/2 + 1 + r$

9. $\bar{h}_{n/2 - r} = h_{n/2 + 1 + r}$ for all $0 \leq r \leq n/2 - 1$ by step 8.

10. Therefore the Hamming vector is anti-symmetric about its center when it is formed from a complementary pair.

* * *

The Hamming vectors of (I, II) and (III, IV) in the sequence quadruple form are invariant under transformation if $H(I, II)$ and $H(III, IV)$ are taken as an unordered pair. The following four theorems which parallel Theorems 4.2, 4.3, 4.4, 4.5 on Hamming distances will demonstrate this characteristic of unordered Hamming vector pairs.

Theorem 5.7

A change in the order of (I, II) or (III, IV) or both will not change $H(I, II)$ or $H(III, IV)$.

1. $H(U, V) = u_i \oplus v_i$ by definition.
2. $u_i \oplus v_i = v_i \oplus u_i$ since modulo 2 addition is commutative,
3. therefore $H(U, V) = H(V, U)$

* * *

Theorem 5.8

Complementing the pair (I, II) or (III, IV) or both will not change $H(I, II)$ or $H(III, IV)$.

1. $H(U, V) = u_i \oplus v_i$ by definition,
2. $u_i \oplus v_i = \bar{u}_i \oplus \bar{v}_i$ by Theorem 2.1,
3. therefore $H(U, V) = H(\bar{U}, \bar{V})$.

* * *

In the following proof it is assumed that each vector I, II, III, IV is of length m . It is also necessary to show that $\bar{\bar{x}} = x$. This is easily seen since $x \oplus 1 = \bar{x}$ and $\bar{x} \oplus 1 = \bar{\bar{x}}$, therefore $\bar{\bar{x}} = x \oplus (1 \oplus 1) = x$.

Theorem 5.9

Complementing one of the pair (I, II) and one of the pair (III, IV) exchanges the Hamming vectors $H(I, II)$ and $H(III, IV)$.

$$I = a_1 a_3 a_5 \dots a_{n-1}$$

$$II = a_n a_{n-2} \dots a_2$$

$$III = b_1 b_3 b_5 \dots b_{m-1}$$

$$IV = b_m b_{m-2} \dots b_2$$

1. $H(I, II) = h_1 h_2 h_3 h_4 \dots h_m$.
2. $H(III, IV) = \bar{h}_1 \bar{h}_2 \bar{h}_3 \bar{h}_4 \dots \bar{h}_m$ because of the parity check.
3. $H(\bar{I}, II) = H(I, \bar{II}) = \bar{h}_1 \bar{h}_2 \bar{h}_3 \bar{h}_4 \dots \bar{h}_m$.
4. $H(\bar{III}, IV) = H(III, \bar{IV}) = \bar{\bar{h}}_1 \bar{\bar{h}}_2 \bar{\bar{h}}_3 \bar{\bar{h}}_4 \bar{\bar{h}}_5 \dots \bar{\bar{h}}_m = h_1 h_2 h_3 h_4 \dots h_m$.
5. Therefore $H(I, II) = H(\bar{I}\bar{II}, IV) = H(III, \bar{IV})$,
6. and $H(III, IV) = H(\bar{I}, II) = H(I, \bar{II})$.

* * *

Theorem 5.10

Exchanging the (I, II) pair with the (III, IV) pair exchanges their Hamming vectors.

1. A change in the Hamming vector can only be caused by a bit change.
2. Exchanging the (I, II) pair with the (III, IV) pair changes position but not bits.

* * *

Earlier in this chapter codes of length 40 generated from the quad were decomposed into kernels of length 10 by use of the Hamming vector. The conjecture was made at that point that perhaps codes of length 200 manufactured from kernels of length 10 could be decomposed into kernels of length 50. In order to prove this decomposition is not possible several more theorems are necessary. These theorems will be proved before proceeding to the decomposition problem.

Theorem 5.11

Under all possible general transformations of a complementary pair where $n > 2$ there are only four possible Hamming vectors, two of which are complements of the other two.

1. (A, B) are a complementary pair of length n where $n > 2$.
2. $H(A, B) = H_o$ interlaced with H_e , where H_o are the odd bits of the Hamming vector and H_e are the even bits of the Hamming vector in reverse order. The symbol $H = H_i * H_j$ will be used for H_o interlaced with H_e in exactly the same manner that (I, II) and (III, IV) are interlaced to form A, B. Also $h_i = a_i \oplus b_i$ as before.

3. Let $H_1 = (I \oplus III) = a_1 \oplus b_1, a_3 \oplus b_3 \dots a_{n-1} \oplus b_{n-1} = h_1 h_3 h_5 \dots h_{n-1}$,
 $H_2 = (II \oplus IV) = a_n \oplus b_n, a_{n-2} \oplus b_{n-2} \dots a_2 \oplus b_2 = h_n h_{n-2} \dots h_2$,
 $H_3 = (I \oplus IV) = a_1 \oplus b_n, a_3 \oplus b_{n-2} \dots a_{n-1} \oplus b_2 = h_1 h_3 h_5 \dots h_{n-1}$,
 $H_4 = (II \oplus III) = a_n \oplus b_1, a_{n-2} \oplus b_3 \dots a_2 \oplus b_{n-1} = h_n h_{n-2} \dots h_2$.

$$4. \quad u_i \oplus v_i = \bar{u}_i \oplus \bar{v}_i \quad \text{by Theorem 2.1,}$$

therefore

$$(\bar{I} \oplus \bar{I} \bar{I}) = H_1 \quad (\bar{I} \bar{I} \oplus \bar{I} \bar{V}) = H_2 \quad (\bar{I} \oplus \bar{I} \bar{V}) = H_3 \quad (\bar{I} \bar{I} \oplus \bar{I} \bar{I}) = H_4.$$

$$5. \quad \text{If } h_i = u_i \oplus v_i, \text{ then } h_i \oplus 1 = \bar{h}_i = u_i \oplus v_i \oplus 1 = \bar{u}_i \oplus \bar{v}_i = u_i \oplus \bar{v}_i$$

therefore

$$H(\bar{U}, V) = H(U, \bar{V}) = (U, V)$$

and

$$\bar{H}_1 = (\bar{I} \oplus \bar{I} \bar{I}) = (I \oplus \bar{I} \bar{I}) \quad \bar{H}_3 = (\bar{I} \oplus \bar{I} \bar{V}) = (I \oplus \bar{I} \bar{V})$$

$$H_2 = (\bar{I} \bar{I} \oplus \bar{I} \bar{V}) = (I \bar{I} \oplus \bar{I} \bar{V}) \quad H_4 = (\bar{I} \bar{I} \oplus \bar{I} \bar{I}) = (I \bar{I} \oplus \bar{I} \bar{I}).$$

$$6. \quad H_1 * H_2 = h_1 h_2 h_3 \dots h_n.$$

$$H_2 * H_1 = h_n h_{n-1} \dots h_1.$$

$$7. \quad \text{But } \bar{h}_{n/2-r} = h_{n/2+1+r} \text{ by Theorem 5.6.}$$

$$8. \quad H_1 * H_2 = h_1 h_2 \dots h_{n/2} \bar{h}_{n/2} \dots \bar{h}_2 \bar{h}_1$$

$$H_2 * H_1 = \bar{h}_1 \bar{h}_2 \dots \bar{h}_{n/2} h_{n/2} \dots h_2 h_1.$$

$$9. \quad \text{Therefore } H_1 * H_2 = \overline{H_2 * H_1} = \bar{H}_2 * \bar{H}_1.$$

$$10. \quad H_3 * H_4 = h_1 h_2 \dots h_n.$$

$$11. \quad H_4 * H_3 = h_n h_{n-1} \dots h_1.$$

$$12. \quad \bar{h}_{n/2-r} = h_{n/2+1+r}.$$

$$13. \quad \text{Therefore } H_3 * H_4 = \overline{H_4 * H_3} = \bar{H}_4 * \bar{H}_3.$$

14. Table 5.1 is an exhaustive list of transformations on a complementary pair in the sequence quadruple form with the Hamming vector for each transformation.

$H_1 * H_2$, $H_2 * G_1$, $H_3 * H_4$, $H_4 * H_3$ and their complements are the only possibilities which exist. Therefore only four Hamming vectors are possible and by steps 9 and 13 they occur in pairs which are complements of one another.

*

*

*

TRANSFORMATION		HAMMING VECTOR	COMMON VECTOR
I	(I II III IV)	$H(I) = H_1 * H_2$	$H(I)$
C_1	(\bar{I} \bar{II} III IV)	$H(C_1) = \bar{H}_1 * \bar{H}_2$	$\bar{H}(I)$
C_2	(I II \bar{III} \bar{IV})	$H(C_2) = \bar{H}_1 * \bar{H}_2$	$\bar{H}(I)$
C	(\bar{I} \bar{II} \bar{III} \bar{IV})	$H(C) = H_1 * H_2$	$H(I)$
A_1	(\bar{I} II \bar{III} IV)	$H(A_1) = H_1 * H_2$	$H(I)$
A_2	(I \bar{II} III \bar{IV})	$H(A_2) = H_1 * H_2$	$H(I)$
T_1	(II I III IV)	$H(T_1) = H_4 * H_3$	$H(T_1)$
T_2	(I II IV III)	$H(T_2) = H_3 * H_4$	$\bar{H}(T_1)$
T	(II I IV III)	$H(T) = H_2 * H_1$	$\bar{H}(I)$
Z	(II \bar{I} \bar{III} IV)	$H(Z) = \bar{H}_4 * \bar{H}_3$	$\bar{H}(T_1)$
Y	(\bar{II} I \bar{III} IV)	$H(Y) = H_4 * H_3$	$H(T_1)$
X	(\bar{II} I III \bar{IV})	$H(X) = \bar{H}_4 * \bar{H}_3$	$\bar{H}(T_1)$
W	(II \bar{I} III \bar{IV})	$H(W) = H_4 * H_3$	$H(T_1)$
V	(\bar{I} II IV \bar{III})	$H(V) = \bar{H}_3 * \bar{H}_4$	$H(T_1)$
U	(\bar{I} II \bar{IV} III)	$H(U) = H_3 * H_4$	$\bar{H}(T_1)$
S	(I \bar{II} \bar{IV} III)	$H(S) = \bar{H}_3 * \bar{H}_4$	$H(T_1)$
R	(I \bar{II} IV \bar{III})	$H(R) = H_3 * H_4$	$\bar{H}(T_1)$
Q	(II \bar{I} IV \bar{III})	$H(Q) = H_2 * H_1$	$\bar{H}(I)$
P	(\bar{II} I \bar{IV} III)	$H(P) = H_2 * H_1$	$\bar{H}(I)$
O	(\bar{II} \bar{I} III IV)	$H(O) = \bar{H}_4 * \bar{H}_3$	$\bar{H}(T_1)$
N	(II I \bar{III} \bar{IV})	$H(N) = \bar{H}_4 * \bar{H}_3$	$\bar{H}(T_1)$
M	(\bar{I} \bar{II} IV III)	$H(M) = \bar{H}_3 * \bar{H}_4$	$H(T_1)$
L	(\bar{II} \bar{I} \bar{III} \bar{IV})	$H(L) = H_4 * H_3$	$H(T_1)$
K	(\bar{I} \bar{II} \bar{IV} \bar{III})	$H(K) = H_3 * H_4$	$\bar{H}(T_1)$
J	(\bar{II} \bar{I} \bar{IV} \bar{III})	$H(J) = H_2 * H_1$	$\bar{H}(I)$
H	(I II \bar{IV} \bar{III})	$H(H) = \bar{H}_3 * \bar{H}_4$	$H(T_1)$
G	(\bar{II} \bar{I} IV III)	$H(G) = H_2 * H_1$	$H(I)$
F	(I \bar{II} \bar{III} IV)	$H(F) = \bar{H}_1 * \bar{H}_2$	$\bar{H}(I)$
D	(\bar{I} II III \bar{IV})	$H(D) = \bar{H}_1 * \bar{H}_2$	$\bar{H}(I)$
π	(II I \bar{IV} \bar{III})	$H(\pi) = \bar{H}_2 * \bar{H}_1$	$H(I)$
B	(\bar{II} I IV \bar{III})	$H(B) = \bar{H}_2 * \bar{H}_1$	$H(I)$
\emptyset	(II \bar{I} \bar{IV} III)	$H(\emptyset) = \bar{H}_2 * \bar{H}_1$	$H(I)$

The E transformation just permutes the first two and last two pairs of (I II III IV) and since modulo 2 addition is commutative no change would take place in the H vector. The last column identifies all Hamming vectors in terms of two vectors and their complements.

TABLE 5.1

Theorem 5.12

If a complementary pair (A,B) of length n is operated upon by a complementary pair (C,D) of length m to form codes U_1, U_2 by the time sequence exponential method, or to form codes V_1, V_2 by the interlace exponential method and either U_1 or U_2 is time reversed, symbolized by \underline{U} or if V_1 or V_2 is time reversed symbolized by \underline{V} then $H(\underline{U}) = H(\underline{V}) = H(A^1, B^1)^{2m}$ where the (A^1, B^1) pair indicates some general transformation of the original (A,B) pair.

1. $\bar{A} \oplus \underline{B} = H(M)$. where B indicates the time reverse of the original B
2. $A \oplus \underline{\bar{B}} = H(H)$.
3. $B \oplus \underline{A} = \underline{A} \oplus B = H(T_1)$.
4. $\bar{B} \oplus \underline{\bar{A}} = \underline{\bar{A}} \oplus \bar{B} = H(L)$.
5. Referring to Table 5.1,

$$H(M) = H(T_1),$$

$$H(H) = H(T_1),$$

$$H(L) = H(T_1).$$

$$6. \begin{aligned} U_1 &= A^{c_1} A^{c_2} \dots A^{c_m} B^{d_1} \dots B^{d_m} \\ U_2 &= A^{d_m} A^{d_{m-1}} \dots A^{d_1} \bar{B}^{\bar{c}_m} \dots \bar{B}^{\bar{c}_1} \end{aligned}$$

and

$$\begin{aligned} V_1 &= A^{c_1} B^{d_1} A^{c_2} \dots A^{c_m} B^{d_m} \\ V_2 &= A^{d_m} \bar{B}^{\bar{c}_m} A^{d_{m-1}} \bar{B}^{\bar{c}_{m-1}} \dots A^{d_1} \bar{B}^{\bar{c}_1}. \end{aligned}$$

Time reversing one of the pair in each, for example, the use of T_2 yields

$$\begin{aligned} U_1 &= A^{c_1} A^{c_2} \dots A^{c_m} B^{d_1} \dots B^{d_m} \\ \underline{U}_2 &= \underline{B}^{\bar{c}_1} \underline{B}^{\bar{c}_2} \dots \underline{B}^{\bar{c}_m} \underline{A}^{d_1} \dots \underline{A}^{d_m} \end{aligned}$$

and

$$\begin{aligned} V_1 &= A^{c_1} B^{d_1} A^{c_2} \dots A^{c_m} B^{d_m} \\ \underline{V}_2 &= \underline{B}^{\bar{c}_1} \underline{A}^{d_1} \underline{B}^{\bar{c}_2} \dots \underline{B}^{\bar{c}_m} \underline{A}^{d_m}. \end{aligned}$$

7. $H(\underline{U})$ and $H(\underline{V})$ consist of clusters of $(A^{c_i} \oplus B^{c_i})$ and clusters of $(B^{d_i} \oplus A^{d_i})$.
8. The Hamming vectors for these clusters have the following four possible forms:

$$\bar{A} \oplus \underline{B}, A \oplus \underline{\bar{B}}, B \oplus \underline{A}, \text{ and } \bar{B} \oplus \underline{\bar{A}}.$$

These four possible vectors were seen in steps 1, 2, 3, 4, 5 to all be equal to $H(T_1)$. If U_1 and V_1 had been time reversed, using the operation T_1 , the result would have been $\bar{H}(T_1)$.

10. Since all the clusters are identical and there is a cluster for each of the m bits in C and each of the m bits in D ,
 $H(\underline{U}) = H(\underline{V}) = [H(T_1)]^{2^m}$ for U_2 or V_2 time reversed or in general

$$H(\underline{U}) = H(\underline{V}) = H(A^1, B^1)^{2^m} \text{ where } (A^1, B^1) \text{ signifies some transformation of } (A, B).$$

* * *

Theorem 5.13

The kernel of length 2 when operated on by a complementary pair by either the time sequence exponential or interlace exponential method can always be decomposed at least once by the interlace method.

1. If one of the code pair formed by either exponential method is time reversed, $H(\underline{U}) = H(\underline{V}) = H(A^1, B^1)^{2^m}$ by Theorem 5.14.
2. There are only two possible Hamming vectors for the quad, 01 and 10.
3. Therefore $H(\underline{U}) = H(\underline{V}) = (01)^{2^m}$ or $(10)^{2^m}$ both of which are decomposable by the interlace method.

* * *

Theorem 5.14

A complementary pair (U_1, U_2) formed by the time sequence exponential method cannot be decomposed by the time sequence method.

1. Let (U_1, U_2) be a complementary pair of length $2mn$ formed from the complementary pair (A, B) of length n operated upon exponentially by the complementary pair (C, D) of length m , formed in the same manner as in Theorem 5.3.
2. $H(U) = (c_1 \oplus dm)^n (c_2 \oplus d_{m-1})^n \dots (c_m \oplus d_1)^n (d_1 \oplus \bar{c}_m)^n \dots (d_m \oplus \bar{c}_1)^n$.
3. The first mn bits in $H(U)$ are m clusters composed of n zeros or n ones dependent upon the bits of $H(C, D)$.
4. By Theorem 5.1 $H = 0^{mn} 1^{mn}$ or $1^{mn} 0^{mn}$, for decomposition of a code of length $2mn$ by the time sequence method.
5. This requires the distance of (C, D) be zero or m , but by Theorem 4.1 $D(C, D) = m/2$, therefore $H(U)$ is not decomposable by the time sequence method.
6. If U_1 or U_2 is time reversed $H(\underline{U}) = H(A^1, B^1)^{2m}$ by Theorem 5.12.
7. This is incompatible with the form $H = 0^{mn} 1^{mn}$ since it would require the first $m/2$ clusters of $H(A^1, B^1)$ to have a distance of zero and the second $m/2$ clusters of $H(A^1, B^1)$ to have a distance of n . Both are impossible by Theorem 4.1 which requires a distance of $n/2$.

* * *

Theorem 5.15

A complementary code pair (V_1, V_2) formed by the interlace exponential method (using the same definitions and symbols as Theorem 5.4) can be decomposed by the time sequence method only if $H(C, D) = 0^{m/2} 1^{m/2}$ or $1^{m/2} 0^{m/2}$.

1. $H(V) = (c_1 \oplus d_m)^{2n} (c_2 \oplus d_{m-1})^{2n} \dots (c_m \oplus d_1)^{2n}$ by Theorem 5.4 and equation 5.5.
2. The clusters of zeros and ones in this Hamming vector are in lengths of $2n$ and are dependent in value upon the bits of $H(C,D)$.
3. $H=0^{mn}1^{mn}$ is the requirement for decomposition of this code by the time sequence method.
4. If $H(C,D)=0^{m/2}1^{m/2}$ or its complement, it will satisfy step 3 since $(0^{2n})^{m/2} (1^{2n})^{m/2} = 0^{mn}1^{mn} = H(V)$, and it also satisfies Theorem 4.1.
5. If V_1 or V_2 is time reversed $H(\underline{V}) = H(A^1, B^1)^{2m}$ by Theorem 5.12.
6. This is incompatible with the form $H=0^{mn}1^{mn}$ since it would require the first $m/2$ clusters of $H(A^1, B^1)$ to have a distance of zero, and the second $m/2$ clusters to have a distance of n . Both are impossible by Theorem 4.1 since it requires a distance of $n/2$.
7. Step 4 showed that $H(C,D)=0^{m/2}1^{m/2}$. This satisfies the requirement for the time sequence decomposition by inspection. Any other arrangement of $H(C,D)$ would fail to allow decomposition.

* * *

Theorem 5.16

Any complementary sequence pair (V_1, V_2) formed by the interlace exponential method as in Theorem 5.4 (using the same definitions and symbols), the quad as the (C,D) pair, is time sequence decomposable.

1. The quad has only two possible Hamming vectors, 01 and 10.
2. This satisfies the form $H=0^{m/2}1^{m/2}$ where $m=2$, the quad length.
3. V_1 and V_2 are therefore time sequence decomposable by Theorem 5.15.

* * *

Theorem 5.17

A complementary pair formed by either exponential method is decomposable by the interlace method if and only if $H(A^1, B^1) = (01)^{n/2}$ or $(10)^{n/2}$. All definitions and symbols are assumed to be the same as those in Theorems 5.3, 5.4 and 5.12.

1. By Theorem 5.2 these codes are decomposable by the interlace method only if they are of the form $H = (01)^{nm}$ or $(10)^{nm}$.
2. $H(U)$ and $H(V)$ are clusters of at least n ones or n zeros and therefore cannot have the form $(01)^{nm}$.
3. $H(U)$ and $H(V) = H(A^1, B^1)^{2m}$ by Theorem 5.12.
4. If and only if $H(A^1, B^1)$ has the form $(01)^{n/2}$ or $(10)^{n/2}$ can $H(U)$ or $H(V) = [(01)^{n/2}]^{2m} = (01)^{nm}$, or using 10 instead of 01, $(10)^{nm}$. This satisfies the requirement of Theorem 5.2 that the Hamming vector be of the form $(01)^r$ or $(10)^r$.

* * *

A brief description of the method used to prove that codes of length 200 formed from kernels of length 10, cannot be decomposed into kernels of length 50, will now be given as a guide to the actual proof of the theorem. The method used is to exhaust all possible methods of decomposition by examining all possible Hamming vectors for the (A, B) and (C, D) pairs. Table 5.2 lists all possible Hamming vectors for kernels of length 10.

Kernel 1	Kernel 2
A=1000000110	A=1001101111
B=1001010001	B=1100001010
The 4 possible Hamming vectors.	The 4 possible Hamming vectors.
0001010111	0101100101
1110101000	1010011010
0000101111	1100101100
1111010000	0011010011

All possible Hamming vectors of kernels of length 10.

TABLE 5.2

The two methods of generating codes of length 200 from kernels of length 10 are the time sequence exponential method and the interlace exponential method given respectively by

$$\begin{aligned} U_1 &= A^{c_1} A^{c_2} \dots A^{c_{10}} B^{d_1} \dots B^{d_{10}} \\ U_2 &= A^{d_{10}} A^{d_9} \dots A^{d_1} B^{\bar{c}_{10}} \dots B^{\bar{c}_1} \end{aligned} \quad (5.6)$$

$$\begin{aligned} V_1 &= A^{c_1} B^{d_1} A^{c_2} B^{d_2} \dots A^{c_{10}} B^{d_{10}} \\ V_2 &= A^{d_{10}} B^{\bar{c}_{10}} A^{d_9} B^{\bar{c}_9} \dots A^{d_1} B^{\bar{c}_1} \end{aligned} \quad (5.7)$$

where (A, B) and (C, D) are kernels of length 10.

The Hamming vectors of these codes of length 200 are anti-symmetric about their centers and are in clusters of 10 zeros or 10 ones for $H(U)$, and in clusters of 20 zeros or 20 ones for the $H(V)$ vector. Time reversing either U_1 or U_2 and similarly either V_1 or V_2 , gives 20 clusters of Hamming vectors of the forms listed in Table 5.2, since each of the 20 A codes is now matched with some transform of B .

The known possible ways for kernels of length 50 to generate composite codes of length 20 are:

1. Either interlace or time sequence kernels of length 50 to form composite codes of length 100 and then either interlace or time sequence these $n=100$ codes to form composite codes of length 200.
2. Take kernels of length 50 and either time sequence exponential or interlace exponential with the quad to form codes of length 200.
3. Take kernels of length 2 and either time sequence exponential or interlace exponential with kernels of length 50.

An examination of all possible Hamming vectors of length 200 from each of these methods will show the incompatibility of these Hamming vectors with those generated by the kernels of length 10.

Theorem 5.18

No codes of length 200 formed from kernels of length 10 can be decomposed into kernels of length 50 by standard methods of decomposition.

1. Given U_1 U_2 and V_1 V_2 both complementary pairs of length formed by the time sequence exponential and interlace exponential methods as given in equations 5.6 and 5.7 respectively.
2. The time sequence method of decomposition is impossible for each of the following Hamming vectors for the following reasons.
 - a. $H(U)$ can never be decomposed by time sequence. Theorem 5.14.
 - b. $H(V)$ requires a form $0^5 1^5$ by Theorem 5.15. Table 5.2 contains no vector of this form.
 - c. $H(\underline{V})$ and $H(\underline{U})$ can never be decomposed by time sequence. Theorems 5.14, 15.
3. The interlace sequence method of decomposition is impossible for each of the vector forms for the following reasons.
 - a. $H(U)$ and $H(V)$ can never be decomposed by interlace. Theorem 5.17.
 - b. $H(\underline{V})$ and $H(\underline{U})$ are decomposable by interlace only if $H(A^1, B^1)$ is of the form $(01)^5$ or $(10)^5$ by Theorem 5.17. Table 5.2 contains no such vectors.
4. Assume that one of the Hamming vectors $H(U)$ or $H(V)$ has a second exponential form, first with the kernel of length 2 operated on exponentially by the kernel of length 50, and secondly by the kernel of length 50 operated on exponentially by the quad. Let H^1 be the designator for the Hamming vectors of the 2 and 50 combinations.
 - a. Consider first the quad acted on exponentially by the kernel of length 50. $H^1(\underline{U}) = H^1(\underline{V}) = H^1(A^1, B^1)$. But since A^1, B^1 is the quad, by Theorem 5.13, it is decomposable by the interlace scheme. This is not possible since step 3 above exhausted all possible interlace decompositions.
 - b. Next consider the kernel of length 50 acted on exponentially by the quad. Considering first the exponential

interlace scheme, by Theorem 5.15 if $H^1(C,D)=01$ or 10 , which it does for the quad. Therefore the code is time sequence decomposable. However, in step 2 all of these time sequence possibilities were exhausted and therefore this decomposition is not possible. $H^1(U)$ would have a form containing either 50 zeros or 50 ones in clusters. For the original 10 length codes to form these would require a $0^5 1^5$ Hamming vector which is not listed in Table 5.2.

This exhausts all possible Hamming vectors that might lead to decomposition, therefore codes of length 200 formed from kernels of length 10 cannot be decomposed into kernels of length 50 by standard methods.

* * *

Table 5.3 shows the possible Hamming vectors for codes of length 26.

A=01001101111010111100111010

B=10110010000111111100111010

The 4 possible Hamming vectors.

11111111111101000000000000

00000000000010111111111111

11101110111010100010001000

00010001000101011101110111

All possible Hamming vectors of the kernel
of length 26.

Table 5.3

An examination of Table 5.3 which is exhaustive for the Hamming vectors of length 26 and Table 5.2 for 10, shows that the next two theorems can be proved by identical methods to those used in Theorem 5.18. Since these proofs exactly parallel that of Theorem 5.18 they will not be given.

Theorem 5.19

Complementary pairs of length 520 formed from the time sequence

exponential or interlace exponential from kernels of length 26 and 10 cannot be decomposed by standard methods into kernels of length 130.

* * *

Theorem 5.20

Complementary pairs of length 1352 formed from the time sequence exponential or interlace exponential method from kernels of length 26 cannot be decomposed by standard methods into kernels of length 338.

* * *

The study of Hamming vectors thus far has disclosed that a pattern of $(01)^{n/2}$ or $0^{n/2} 1^{n/2}$ or their complements in either the (A,B) pair or (C,D) pair, is required in order that a code formed from one kernel length by an exponential method be decomposed into a pair from a different kernel length. The quad is only kernel which thus far satisfies this criteria. However, the other kernels come close, as a check shows that $0^4 101^4$ and $(01)^2 10(01)^2$ are among the Hamming vectors of length 10, and $0^{12} 101^{12}$ is a Hamming vector of length 26.

If the quad is involved in the formation of composite codes it makes their decomposition more likely. This is shown in Theorem 5.12 and Theorem 5.16. Appendix II demonstrates this fact quite clearly since it contains all codes of length 16, less some operational redundancies. Each of the code pairs and its time inverse is decomposable by some method into shorter length codes. Appendix III lists all the pairs of length 20, less some operational redundancies. Each code pair or its time inverse can be broken into its originating pair through the use of Hamming vectors, but no code and its time inverse can both be decomposed by the standard Hamming vector methods.

The writer would like to end this chapter with a conjecture that kernels of length 50 do not exist. This conjecture is based upon three

bits of evidence, the proof of Theorem 5.18 being the first. The second piece of evidence is based upon the fact that $50=7^2+1^2=5^2+5^2$. From this one might assume that codes of 200 formed from 10's might cover just one set of the ones determined by equation 2.5. This proved not to be true however, since the number of ones in both the exponential formations of codes of length 200 from kernels of length 10, if they were decomposable into kernels of length 50, would exhaust all possible unordered pairs of ones for $n=50$ as determined by equation 2.5. The last bit of evidence to base this conjecture upon is that the code length is not twice a prime number. The only previous kernel which might have existed, but was not twice a prime number, was $n=18$, and it did not exist. Similar reasoning extends this conjecture to kernels of length 130 and 338.

CHAPTER VI

SUPPLEMENTARY AND CYCLIC COMPLEMENTARY CODES

This chapter is concerned with two classes of codes. The first of these classes consists of quadruples of sequences with the property that the total number of likes at each spacing equals the total number of unlikes at the same spacing. These are called supplementary codes.² The second class of codes, which will be defined later, consists of the cyclic complementary codes. Both of these code types have complementary sequence pairs as a possible subset; this allows the use of their properties as screens in the search for new kernels.

Supplementary codes will be discussed first, since the $n=26$ search involved the use of this property as a screen, while the cyclic complementary property was not used until the search for kernels of length 34. Only one theorem will be proved for each of these types of codes, since these two theorems were the only ones applied in the actual search for new kernels.

If
$$\begin{array}{l} u_1 u_2 u_3 \dots u_n \\ v_1 v_2 v_3 \dots v_n \\ w_1 w_2 w_3 \dots w_n \\ x_1 x_2 x_3 \dots x_n \end{array}$$
 are sequences of zeros and ones and

satisfy the constraint of equation 6.1, they form a quadruple which is supplementary.

$$\begin{aligned} & \sum_{i=1}^{i=j} (u_i \oplus u_{n+i-j} \oplus 1) + (v_i \oplus v_{n+i-j} \oplus 1) + (w_i \oplus w_{n+i-j} \oplus 1) + (x_i \oplus x_{n+i-j} \oplus 1) = \\ & \sum_{i=1}^j (u_i \oplus u_{n+i-j}) + (v_i \oplus v_{n+i-j}) + (w_i \oplus w_{n+i-j}) + (x_i \oplus x_{n+i-j}) = 2j \end{aligned} \quad (6.1)$$

for all j , $1 \leq j \leq n - 1$. Note that there are $2j$ like pairs and $2j$ unlike pairs at a spacing of $n-j$.

For simplicity in the proof, we use the symbolism that was used in Chapter 2,

$$\text{let } L_U = \sum_{i=1}^j u_i \oplus u_{n+i-j} \oplus 1 \quad \text{for all } j, 1 \leq j \leq n-1$$

$$U_U = \sum_{i=1}^j u_i \oplus u_{n+i-j}$$

A restatement of equation 6.1 in this symbolism is

$$L_U + L_V + L_W + L_X = U_U + U_V + U_W + U_X \quad (6.2)$$

Theorem 6.1

Any complementary sequence pair, (A,B), written in standard (I II III IV) form has the property that I,II, III and IV are a supplementary quadruple.

1. $L_A = U_B$ and $U_A = L_B$ by the definition of complementary.
2. Considering just the even values of j rather than all values,

$$L_A = L_I + L_{II},$$

$$U_A = U_I + U_{II},$$

$$L_B = L_{III} + L_{IV},$$

$$U_B = U_{III} + U_{IV}.$$

$$3. \quad L_I + L_{II} = U_{III} + U_{IV}.$$

$$4. \quad L_{III} + L_{IV} = U_I + U_{II}.$$

5. Adding the equations in steps 3 and 4 gives

$$L_I + L_{II} + L_{III} + L_{IV} = U_I + U_{II} + U_{III} + U_{IV}.$$

6. This is the same as equation 6.2 and I, II, III, IV formed in the sequence quadruple form from a complementary pair are supplementary.

Since the supplementary property applies only to the even spacings in a code pair, the converse, that all interlaced supplementary quadruples are complementary, is not true.

The cyclic complementary property is useful in itself for a communications or telemetering system; however, at this time our concern is for its property as a necessary condition for a code pair to be complementary. A cyclic sequence or code, as the name suggests, is a never ending sequence of zeros and ones which has a period of n bits. A cyclic complementary sequence pair, is a pair of cyclic codes, each of period n , where the number of likes of one sequence equals the number of unlikes of the other sequence for n possible matches for all spacings from 1 to $n-1$, or stated in terms of the bits of the A and B codes

$$C_j = \sum_{i=1}^n a_i \oplus a_{n-j+i} = \sum_{i=1}^{i=n} b_i \oplus b_{n-j+i} \oplus 1 \quad 1 \leq j \leq n-1. \quad (6.3)$$

where $a_{n+i} = a_i$ and $b_{n+i} = b_i$ since the period is n .

Theorem 6.2

A complementary sequence pair (A, B) of length n , if written in a cyclic fashion, is always a cyclic complementary pair of period n .

1. Since (A, B) are a complementary pair they satisfy equation 2.3.

$$f_j = \sum_{i=1}^j a_i \oplus a_{n-j+i} = \sum_{i=1}^{i=j} b_i \oplus b_{n-j+i} \oplus 1 \quad \text{for } j, 1 \leq j \leq n-1.$$

2. Expanding for f_1 gives

$$a_1 \oplus a_n = b_1 \oplus b_n \oplus 1 \quad \text{which is equal to } a_n \oplus a_1 = b_n \oplus b_1 \oplus 1.$$

3. Expanding for f_{n-1} gives

$$(a_1 \oplus a_2) + (a_2 \oplus a_3) + (a_3 \oplus a_4) + \dots + (a_{n-1} \oplus a_n) = (b_1 \oplus b_2 \oplus 1) + (b_2 \oplus b_3 \oplus 1) + (b_3 \oplus b_4 \oplus 1) + \dots + (b_{n-1} \oplus b_n \oplus 1).$$

4. Adding the equations of step 2 to step 3 gives C_1 .

5. Similarly expanding f_2 gives

$$(a_1 \oplus a_{n-1}) + (a_2 \oplus a_n) = (b_1 \oplus b_{n-1} \oplus 1) + (b_2 \oplus b_n \oplus 1)$$

which is equal to

$$(a_{n-1} \oplus a_1) + (a_n \oplus a_2) = (b_{n-1} \oplus b_1 \oplus 1) + (b_n \oplus b_2 \oplus 1).$$

6. Expanding f_{n-2} gives

$$(a_1 \oplus a_3) + (a_2 \oplus a_4) + (a_3 \oplus a_5) + \dots (a_{n-2} \oplus a_n) = (b_1 \oplus b_3 \oplus 1) + (b_2 \oplus b_4 \oplus 1) + (b_3 \oplus b_5 \oplus 1) + \dots + (b_{n-2} \oplus b_n \oplus 1)$$

7. Adding the equations of step 5 to step 6 gives C_2 .

8. Continuing this same procedure through all possible values of j would show that $C_j = f_j + f_{n-j}$. Therefore the cyclic complementary constraint equations are based on the sum of two restricted portions of the complementary constraint equations. Therefore all complementary pairs have to be cyclically complementary. The converse is not necessary true since C_j is a sum, and in a sum the addend and augend are not unique.

* * *

Since $C_j = f_j + f_{n-j}$, each constraint of the complementary property is used twice as j varies from 1 to $n-1$; therefore C_j is symmetric and is centered at $C_{n/2}$. In the application of the cyclic complementary property in Chapter 8, half of the characteristic cyclic number is deleted due to this symmetry property.

Both of the screens described in this chapter were suggested by Dr. Golay;^{14, 19} however, to the best of my knowledge the theorems and their proofs have not appeared in print.

AN EXHAUSTIVE SEARCH FOR KERNELS OF LENGTH 26

CHAPTER VII

In his paper Golay emphasized the need for an exhaustive search for complementary code pairs of length 26. This chapter describes such a search. The purpose of this search was to determine if there were any kernels of length 26 and if any of these exist, the number of such kernels. It would prove very difficult to surmise the number of kernels in a code of this length predicated on the known kernels of other lengths, since $n=2$ had one kernel, $n=10$ had two, and $n=18$ had zero. An exhaustive search for kernels of length 26 might throw some light on a possible general method for finding kernels of longer lengths, or might disclose some sort of pattern showing the distribution of kernels among the possible code lengths.

The only feasible method to accomplish this search was with a high speed digital computer. This chapter is concerned with the computer program and the results of the exhaustive search for kernels of length 26.

At first glance it would appear that there are 2^{52} possibilities to be screened for an exhaustive search for all possible kernels of length 26. Although this is true, the application of some of the theorems developed in the earlier chapters immediately eliminates from consideration large blocks of the 2^{52} possibilities which are either redundant or impossible. This is a very necessary procedure since even with the highest speed computers of today it would be impossible to investigate this number of possibilities in a life time.

The program was developed for the CDC 1604; this computer accomplishes approximately 200,000 operations per second. This speed coupled with the powerful screens used in programming made possible a reduction of the computer run-time to approximately 75 hours. A brief outline of the various screens used and their reduction factor will now be

given before continuing in more detail with the actual programming techniques used. The word code or code pair as used in this chapter and also in the next chapter will be understood as possible complementary code pair.

The first screen utilized was the number of ones which must appear in each of the code pairs. This was given by $n = (n-p-q)^2 + (p-q)^2$ as derived in chapter 2, where n is the length of the code, p is the weight of the A code and q is the weight of the B code. The possible solutions for $n=26$ were the unordered weight pairs (16,15), (16,11), (11,10), (15,10). The pair ($p=16$, $q=15$) was arbitrarily chosen to be used in the program. This reduced the total number of possible code pairs from $2^{52} \approx 8 \times 10^{15}$ down to $\frac{26!}{16!10!} \times \frac{26!}{15!11!} \approx 4 \times 10^{13}$. This was a reduction by a factor of 200, which brought the life time search down to a little less than a year, but this amount of time was still not feasible for a computer search. Golay remarked in a footnote that a complementary pair might be thought of as being composed of two interlaced half length codes, in this case $n=13$.⁹ Since it was obvious the total number of ones in both of the interlaced pair must equal the number of ones in one of the complementary code pair, equation 2.5 was applied again, this time for $n=13$. The unordered weight possibilities for $n=13$ were (9,7), (9,6), (6,4), (7,4). In order to be compatible with the (16,15) pair for $n=26$, the (9,7) pair was selected for the code containing 16 ones and the (9,6) pair was chosen for the other code which contained 15 ones.

The basic problem had now been reduced to all possible combinations of the four thirteen bit sequence families. This was equal to

$\frac{13!}{9!4!} \times \frac{13!}{9!4!} \times \frac{13!}{7!6!} \times \frac{13!}{7!6!} = 715 \times 715 \times 1716 \times 1716$ or approximately 1.5×10^{12} possibilities. Although this reduced the problem by one more

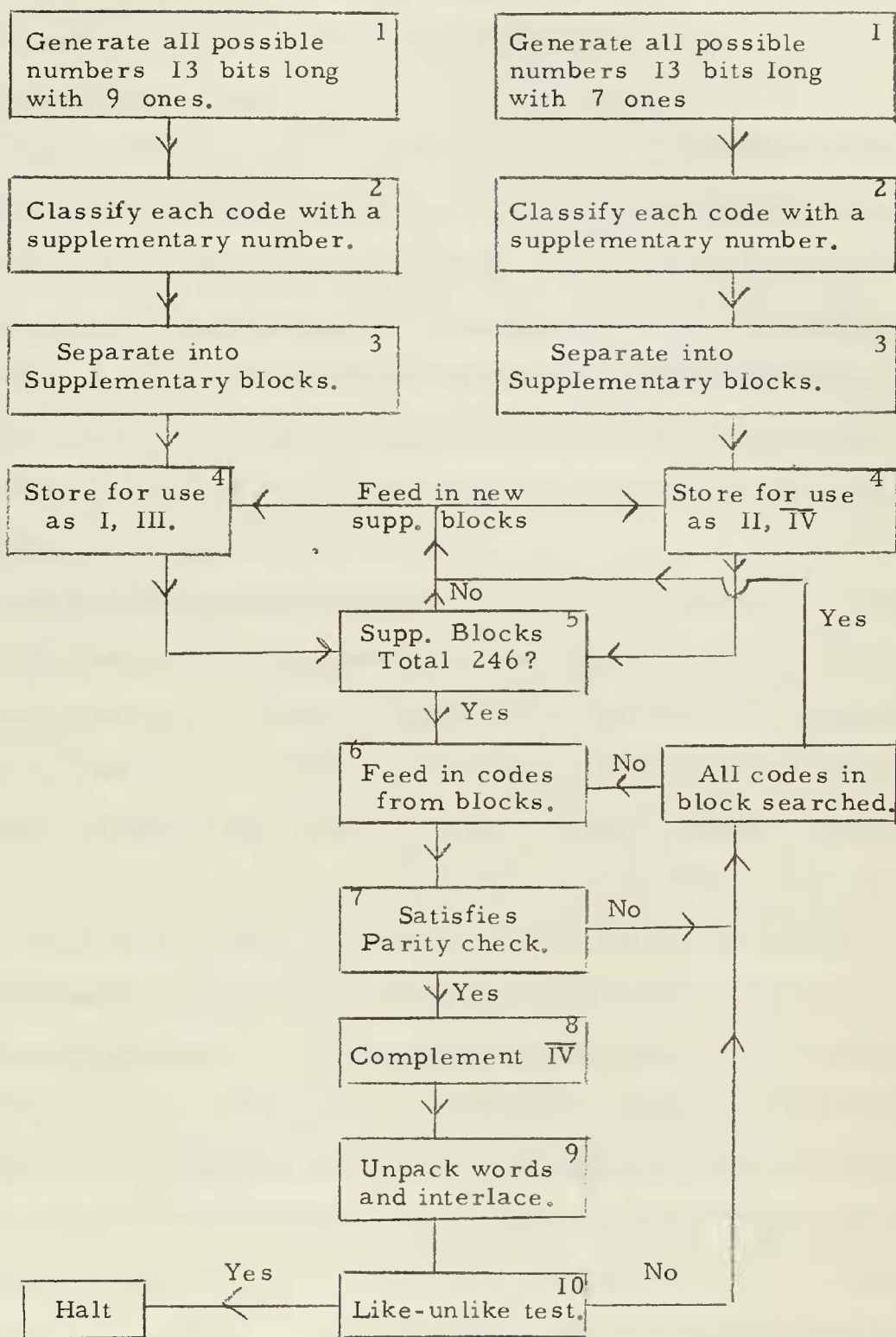
order of magnitude it was still much too large for a practical search. Dr. Golay in a private communication with the author pointed out the possibility of using the supplementary characteristic as an additional screen.¹⁴ Theorem 6.1 gives the necessary and sufficient condition that complementary codes sequence quadruple form be supplementary sequence quadruples. The codes of length 13 were therefore categorized according to their first 3 and last 3 bits. The number of like pairs at spacing twelve, at spacing eleven, and at spacing ten was computed for each of the 13 bit sequences. For example, 0111100111011 has 0 likes for spacing twelve, 1 like for spacing eleven, and 3 likes for spacing ten. The block number given to 0111100111011 was therefore 013. Out of sixty four possible combinations only eighteen different block numbers were generated for each of the two number sequences of all possibilities of thirteen bits with nine ones and with seven ones. An examination of Figure 7.3 reveals that there are a widely varying number of members within each of the blocks. The supplementary characteristic states that the total number of like pairs for each spacing must equal the total number of unlike pairs for the same spacing. With four codes at a spacing of twelve there are 4 possibilities, therefore two of them must be likes; at a spacing of eleven there are eight possibilities, therefore four of them must be likes; and likewise at a spacing of ten there are twelve possibilities of which six must be likes to satisfy the supplementary property. Block numbers were added for each possible combination to see if the total were 246, and in cases where the total was 246 those particular codes within these blocks were then sent to the next necessary but not sufficient condition for the codes to be complementary. If the total was not 246, the blocks of codes were rejected. The power of this screen is shown in the example where say two of the blocks had code

numbers 000, 000. The only possible combination to add to 246 would be 123, 123 for the other two codes. This cuts the total number of possibilities in this particular block combination down from $1716 \times 1716 = 2.9 \times 10^6$ to only 64 possibilities. Although this admittedly was the most extreme example, this screen reduced the number of possible codes checked to about 2×10^9 or further reduction of three orders of magnitude.

The next necessary condition to be checked was the parity test, equation 2.4. All code quadruples which satisfied the parity test were then sent to the necessary and sufficient like pair, unlike pair check for each spacing. The like, unlike check subroutine was quite long and also involved word unpacking. This subroutine which was ignored in the calculation of run time increased the computer search time from a calculated 70 hours to an actual run time of the order of 75 hours.

The half length codes in the program were interlaced in the same manner as was done in the operations group formulation. This method of combination made the parity check easy to calculate since the bit positions in the computer words were the same as the sequence positions of the codes. Modulo 2 addition was directly applied to the code quadruple to form a vector of all ones, providing the parity test held. If the code vector was not all ones the code quadruple was rejected.

It is to be noted that by selecting the ordered (16, 15) pair for the number of ones in the A and B code, the operations C_1 , C_2 , and C were eliminated from possible redundant consideration. These were eliminated by their requirement for (10, 15), (16, 11), and (10, 11) ordered pairs of ones respectively, based on equation 2.5. The interlacing of the half length codes with ordered pairs of (9, 7) ones and (9, 6) ones respectively eliminates from consideration the operations



Flow Chart for Computer Search
For Kernels of Length 26.

FIGURE 7.1

T_1 , T_2 , T and A_1 since these would have required $(7,9)$, $(9,6)$, $(9,7)(6,9)$, $(7,9)(6,9)$ and $(4,7)(4,6)$ ones respectively. The only basic transformation not deleted by controlling the weight in each of the code quadruples was A_2 . This is seen to be true because if the pair $(9,7)(9,6)$ has the operation A_2 performed on it, the result is $(9,6)(9,7)$. If this code is now transformed by the trivial exchange operation, E , a code pair of the original form is generated. The scheme which was used to eliminate the A_2 redundancy will be discussed late in this chapter.

Figure 7.1 is a rough flow diagram for the program used in the exhaustive search for kernels of length 26. With modifications in data as necessary, this program was also used for exhaustive searches of composite codes of 16 and 20.

A brief description of the blocks in Figure 7.1 will give the reader some familiarity with the programming philosophy used in this search. The actual programs as written for the CDC 1604 are given in Appendix V.

The number generation subroutine is able to generate any number up to 48 bits in length (word size of the 1604) with any number of ones up to a maximum of 48. This is accomplished in the 1604 by inserting a number of the form 00000...00001111...111 into the accumulator, where the total length of the number is 48 bits and it contains p ones and has a code length of n . This number is then shifted left $48-n$ bits. This shifted result is then checked to see if it is negative, which indicates a one in the most significant bit position. A procedure is now set up to count a one or zero and then shift left one bit and repeat the process for the entire n bits. If p ones are counted the number is stored, if not it is rejected. A number one larger than its predecessor is inserted into the system and the process is repeated until the ones are all in the uppermost bit positions.

Blocks 2,3,4 are used to compute a supplementary characteristic number for each of the 715 thirteen bit numbers with 9 ones and also for each of the 1716 thirteen bit code numbers with 7 ones; then in accordance with their supplementary numbers these are stored into group blocks. The supplementary number was derived by using a mask to expose for consideration only the first 3 and last 3 bits of each of the 13 bit code numbers. These masked numbers were then compared with an exhaustive list of all possible combinations of zeros and ones in these 6 bits. These 64 possible combinations were in an M to one correspondence with a list of 18 possible supplementary characteristic numbers (see Figure 7.2). This correspondence allowed the temporary attachment of this supplementary number to each of the codes. The code numbers were rearranged into blocks according to supplementary number and at the same time had their temporarily attached supplementary number deleted. This part of the program is purely for putting the data into a useable state and is used only once per run. The rest of the program is highly iterative.

Consideration of the (I II III IV) form of the possible 26 bit codes, shows that each of the 18 blocks of I, II, III, and IV must be compared against each other indicating $18 \times 18 \times 18 \times 18 \approx 105,000$ possible block comparisons. This was cut down somewhat by use of a slightly different technique. This was accomplished by loading the computer accumulator with 246 and from this subtracting the first block number of I, then subtracting the first block number of III, and then subtracting the first block number of II. Rather than subtracting the Block number of IV, the difference obtained was instead checked against all all possible block numbers of IV to see if it was listed. Since if the block number did exist only one could exist. This procedure cut the

possible number of block comparisons by a factor of eighteen and left only 5800 combinations to examine rather than 105,000. If the block did exist all the codes within this (I II III IV) block grouping were then given the parity test. If the IV block number did not exist II was stepped ahead one block number and the process repeated. After II had cycled through its 18 block numbers, III was then stepped ahead one block number and the entire process repeated. Similarly when III had cycled its 18 block numbers, I was stepped ahead and the process repeated. To avoid the A_2 operation redundancy, whenever I was stepped up one block number III was started from this same block number rather than at the first block number. This avoided the block number combinations (I III), and (III I) both appearing except when block I was equal to block III. This block number cycling allowed a vantage breakpoint for partitioning the program into suitable size increments for computer run times.

The parity check was made by adding (Modulo 2) a code from I to a code from III to a code from II. This resultant was then checked to see if it existed as a code in the list from the IV block. If it did exist it was complemented and then sent to be unpacked for the necessary and sufficient like, unlike test. If it did not exist another set of codes was sent in and the process repeated until all possible code combinations in (I II III) were exhausted.

The methods used in the unpacking and like-unlike subroutines are quite straight forward and will not be amplified here, although it is to be noted that the like-unlike subroutine is not limited in the lengths of the codes it can test, up to the machine storage size, whereas all the other subroutines, since they use packed words, do have quite restrictive code length limitations.

This program was first used to make an exhaustive search for codes of length 10 for test purposes since all kernels of this length were known. After this satisfactory checkout the exhaustive search for codes of length 26 was undertaken.

During the first part of the computer run for $n=26$ Dr. Golay informed the writer by personal correspondence that he had discovered a kernel of length 26, using a "by-hand" technique.^{21,20} The exhaustive search revealed only a transformation of the code pair that Dr. Golay had discovered. Therefore it was proved that only one complementary kernel of length 26 existed and that it was (ignoring allowable transformations)

A=01001101111010111100111010

B=10110010000111111100111010 .

Supplementary Characteristic Number	Number of 13 bit codes with nine ones	Number of 13 bit codes with seven ones
0	14	70
20	14	70
110	28	140
1	44	112
11	44	112
21	44	112
101	22	56
111	88	224
121	22	56
2	14	70
12	98	196
22	14	70
102	28	140
112	70	56
122	70	56
13	44	112
103	22	56
123	35	8
Total	715	1716

Number of codes within each
supplementary block for $n=26$

Figure 7.2

123		001		111Cont.		102	
000 000	0	110 000	14000	000 110	6	101 001	12001
111 111	16007	000 011	3	111 001	16001	100 101	10005
012		111 100	16004	110 101	14005	011 010	06002
		001 111	02007	101 011	12003	010 110	04006
100 000	10000	011		100 111	10007	022	
000 001	1	101 000	12000	103		100 110	10006
110 010	14002	000 101	5	010 010	04002	011 001	06001
101 100	12004	111 010	16002	101 101	12005		
010 011	04003	010 111	04007	121			
001 101	02005	013		001 100	02004		
111 110	16006	100 100	10004	110 011	14003		
011 111	06007	001 001	02001	000			
112		110 110	14006	111 000	16000		
		011 011	06003	000 111	7		
010 000	04000	021		002			
000 010	2	100 010	10002	110 100	14004		
111 101	16005	010 001	04001	001 011	02003		
101 111	12007	101 110	12006	110			
122		011 101	06005	110 001	14001		
		101		100 011	10003		
001 000	02000	100 001	10001	011 100	06004		
000 100	4	011 110	06006	001 110	02006		
111 011	16003	111		020			
110 111	14007	011 000	06000	101 010	12002		
		010 100	04004	010 101	04005		
		001 010	02002				

Mask of digits for likes, actual numbers for n=26.

FIGURE 7.3

CHAPTER VIII

A PARTIAL SEARCH FOR KERNELS OF LENGTH 34

The next possible kernels for investigation after length 26 were those of length 34. As will be shown in this chapter an exhaustive search for kernels of length 34 was not feasible. However, an important subset of possible kernels was exhaustively searched and a scheme which can be adapted to the general search was programmed. No kernels were found in this partial search. To use the same method of attack as that used on $n=26$ did not seem too feasible, both because of the length of time involved in the search and because there would be no guide to searching, beyond random picking of possibilities until all possibilities were exhausted. The length of time involved for an exhaustive search of $n=34$ was estimated fairly reasonably by checking the number of possible codes of length 26 against the number of possible codes of length 34, since the time for the 26 search was known with reasonable accuracy. For $n=26$ there were $715 \times 715 \times 1716 \times 1716 \approx 1.5 \times 10^{12}$ code possibilities at one point in the screening process. At the same point in the screening process for $n=34$ codes there were $12376 \times 12376 \times 19448 \times 19448 \approx 5.8 \times 10^{16}$ possibilities, which is a ratio of 4×10^4 to one as compared to $n=26$. A time estimate from the 26 length code to the 34 length code was therefore $75 \times 4 \times 10^4 = 3 \times 10^6$ hours. This length of time was of course not practical for a search of an exhaustive nature.

The use of the cyclic complementary property as a screen was pointed out by Dr. Golay as highly practical because some reasonable guess could be made as to which cyclic sets might contain a kernel.¹⁹ This would lead to a hybrid "by-hand" and computer search. It would also be easier to document the areas searched by the cyclic method as compared to the supplementary method and would therefore avoid the

duplication of effort by future $n=34$ length searchers. As will be discussed at some length, there is in addition an even more important reason leading to the decision to substitute the cyclic complementary property for the supplementary property as a screen.

The generation of cyclic sets which are to be utilized in the search for new kernels, is accomplished in a somewhat unusual fashion. This method of classifying, puts within the same set all possible numbers which are cyclic permutations of one another, as is to be expected; however, there are also contained within the same set all numbers which are formed by the removal of every other bit in a cyclic fashion until all bits are used. As an example the five bit number 11010 will be used to show all the numbers contained within its set. First, selecting every other bit, starting with the first bit, and repeating on

the result gives:

1. 11010
2. 10011
3. 10101
4. 11100
1. 11010

and second, by permuting the bits of each of these in the normal cyclic permutation manner yields

1. 11010, 01101, 10110, 01011, 10101
2. 10011, 11001, 11100, 01110, 00111.

It is not necessary to generate the cyclic permutations from 3 and 4 since they are contained within 1 and 2.

Counting cyclically the number of likes for each spacing gives the code number 1331 for 1 and the code number 3113 for 2. These code numbers start with a spacing of 1 and include up through a spacing of $n-1$ or four in this particular example. The code number 3113 means that at a spacing of one there were 3 likes in 10011 counted cyclically, at a spacing of two there was 1 like, at a spacing of three

there was 1 like, and at a spacing of four there were 3 likes. The example just given is somewhat trivial since all binary numbers of length 5 with 3 ones fall into the same set. However, this is not the case for $n=26$ as there are 7 sets for codes of length 13 with 9 ones and 76 sets for codes of length 13 with 7 ones.

An examination of the kernel of length 26, shows that its sequence quadruple of codes came not from four different cyclic sets as one would expect but came instead from just two cyclic sets.

Using the kernel form

A=01001101111010111100111010

B=10110010000111111100111010 one can decompose this into

I=0010111110111

II=0010110011101

III= 1 101001110111

IV= 0 010111100010

Using I to form its cyclic set gives:

1. 0010111110111
2. 0111111001101
3. 0111011111010
4. 0101100111111
1. 0010111110111

Now take 3 and write it twice, this gives 01110111(1101001110111)11010. Picking off from the ninth to the twentieth second bit gives a code which is the same as III above. Therefore I and III are members of the same cyclic set. Similarly take II and form its cyclic set; this gives:

1. 0010110011101
2. 0110111001010
3. 0111000101011
4. 0100001110111
5. 0001111100101

6. 0011011011100
7. 0101110011010
8. 0010100111011
9. 0110101000111
10. 0111011100001
11. 0101001111100
12. 0001110110110
1. 0010110011101

To make a comparison between II and IV the complement of IV will have to be used in order that it will have 7 ones as does II.

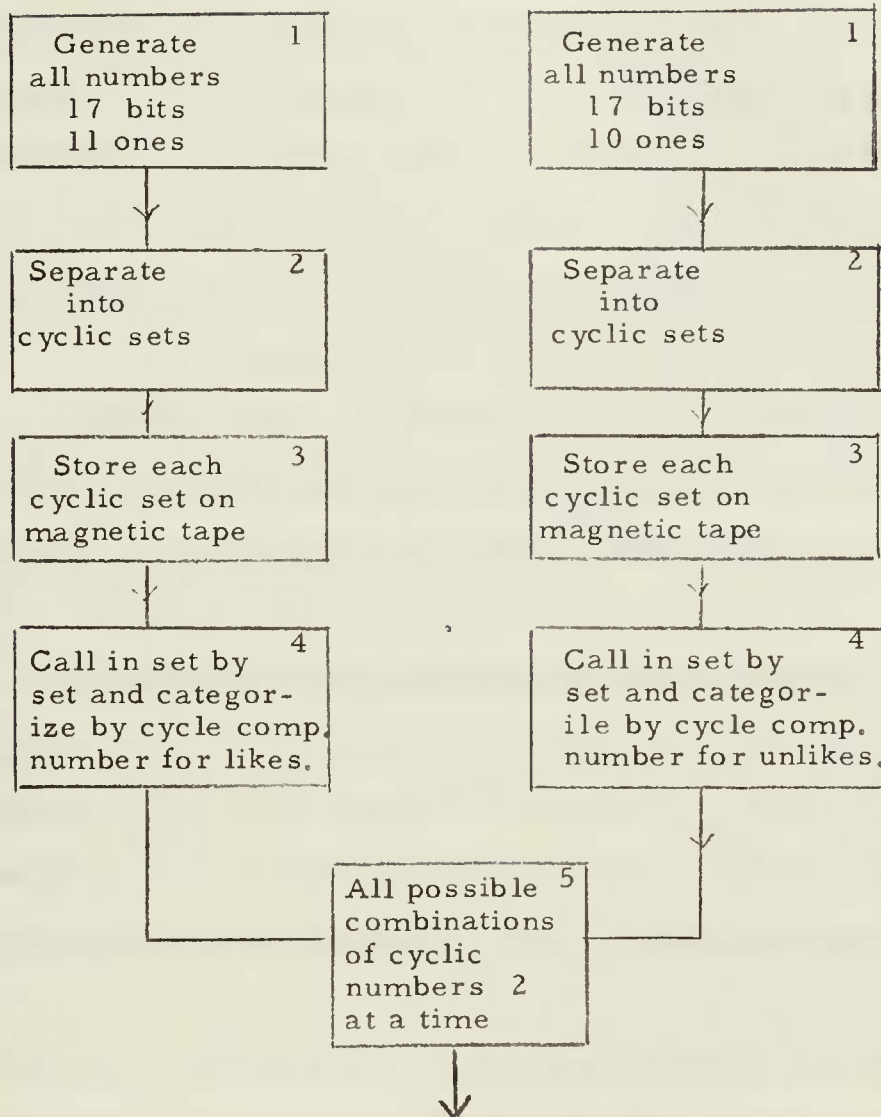
$$\overline{IV} = 1101000011101$$

Writing 4 from this list of 12 twice gives 01000011101(1101000011101)11. Picking off from the twelfth to the twenty fifth bit gives \overline{IV} , which indicates that II and IV are from the same cyclic set.

A check of kernels of length 10 to see if their quadruples came from cyclic sets in pairs, as the 26 kernel did, proved that both kernels of length 10 did in fact come in pairs. This was meaningless however, since as was seen in the example of length five with three ones, there was only one cyclic set possible for each pair.

In the hope that if the kernel existed in the $n=34$ case, it would also be formed with I, III from one cyclic set and II, \overline{IV} from another cyclic set, the program was modified. This change along with giving an exhaustive search for code possibilities taken two at a time from cyclic sets would also give an estimate of the computer run time for the exhaustive search in general using the cyclic complementary property. This was the primary reason for the change in the program.

Eight octal digits were used to designate the cyclic complementary number for each of the 12,376 possible half codes of length 17 with 11 ones and for each of the 19,448 possible half codes of length 17



Flow Chart for
Cyclic Complementary Designation.

FIGURE 8.1

with 10 ones. Only eight digits were necessary as each cyclic count of spacing i actually counts both the i spacing and the $n+1-i$ spacing simultaneously. This was demonstrated by step 8 in the proof of Theorem 6.2. The example of the code of length five given earlier in this chapter showed this symmetric property, as the code numbers were 1331 and 3113. The first two digits of these cyclic complementary designators contain all the information available.

The revision made to the computer program was not too extensive as only blocks 2,3,4,5 in Figure 7.1 were modified to handle the cyclic complementary property rather than the supplementary property. Of course all blocks were modified to handle codes of length 34 rather than codes of length 26.

One difficulty encountered was the lack of computer memory for the large blocks of data, and it was necessary to store the cyclic sets on magnetic tape rather than in the main memory core as the 26 case was handled. The cyclic sets were then called in two at a time for checking. This slowed the search down a little, but by no more than 1% of the total time. Figure 8.1 is a block diagram of the process of obtaining the cyclic complementary numbers and it will be worthwhile to remember the memory size restriction when reviewing the procedure used.

A general description of Figure 8.1 will show the differences in the method used in the search for $n=34$ kernels as compared to that used for $n=26$ kernels. The procedure in the $n=34$ data formation was identical with that of $n=26$ as far as the initial possible number generation was concerned. (Block 1 in Figures 7.1 and 8.1). There was an immediate departure from the old method in blocks 2,3, and 4 where one of the 17 bit numbers with 11 ones was shifted left 17 bits and then added to itself. This formed a 34 bit number which was

actually the 17 bit number written twice in time sequence. A mask was then put over the last 17 bits and this masked number was used, with an equality search of the 12,376 generated 17 bit numbers, to identify this particular number in the list. After the number equal to the reference number was located in the generated list it was tagged a one, indicating cyclic set one. The whole 34 bit number was then shifted right one bit and a mask used to again pick off the last 17 bits. The search through the list and tagging with the group number was repeated. After cycling through all 17 possible cyclic codes and tagging each, the original 34 bit number was now repeatedly long right shifted one bit and then right shifted one bit. A long right shift saves the bit which is pushed off the A register and stores it in the Q register, while the right shift just pushes the extra bit off the end of the A register. Therefore every other bit of the 34 code was saved in Q and this formed a new 17 bit code within the same complementary cyclic set. This code was then formed into a double length code and the list of all possible 17 bit numbers with 11 ones was searched and tagged with a one and the process which was first described to generate cyclic permutations was repeated. This process of first forming 17 cyclic permutations of a code and then taking every other bit, was continued until the possible code which was originally operated upon reappeared. Then cyclic set 2 was started and tagged accordingly, and so on through 98 sets. The 19,448 codes of length 17 with 7 ones were classified in exactly the same manner and formed 150 cyclic complementary possibility sets.

After each series of sets were formed a mask was used to search the set tag numbers and pick off each set for transfer to magnetic tape. To insure that the pick-off was exhaustive, a zero was inserted on the list in place of each removed tagged number.

The sets were now ready to be called back into the computer for use in the search. Normally only one set was called in at a time because the pair from the I, III set was checked against all the II, IV sets before a new I, III set was called into the computer. The I, III sets contained the 11 weight codes while the II, IV set contained the 7 weight codes. When a set was called into the computer each cyclic permutation subset of 17 possible codes was given a cyclic complementary number to categorize the entire subset. Each group contained either 4 or 8 of these cyclic subsets. These subset numbers were formed by counting the likes for the I, III codes and the unlikes for the II, IV codes for the eight unique spacings. The codes within the subsets were checked against each other only if the likes equaled the unlikes of the cyclic complementary subset numbers for all spacing, or $L_I + L_{III} = U_{II} + U_{IV}$.

Figure 8.1 is joined to Figure 7.1 just beyond block 5 and takes the place of blocks 1,2,3,4,5 in Figure 7.1. The same feedback paths shown in Figure 7.1 to insure the checking of all codes within a set and to feed in a new set after a check has been completed, are still in operation for the same purpose. Appendix VI contains the computer program for Figure 8.1.

The total run time for the $n=34$ search, taking I, III from the same cyclic set and II, IV from another cyclic set was approximately 15 hours. No codes were found. An extrapolation from this run time to the exhaustive run time for $n=34$ gives $15 \times 150 \times 98 \approx 2.2 \times 10^5$ hours, which although one order of magnitude less than the supplementary method, is still not reasonable.

A second partial search for $n=34$ kernels was attempted after observing that the kernel of length 10

A=0101000011

B=0000100110

with its center quad removed yields

$A^1=01010011$

$B^1=00000110$

which is a complementary pair of length 8. Breaking A^1, B^1 into its (I II III IV) configuration gives

I=0001

II=1011

III=0001

IV=0100

where (I, II) and (III, IV) are both complementary pairs.

An attempt was made to draw an exact parallel from this observation to codes of length 34. One possible solution to equation 2.5 which determines the number of ones necessary in each of the kernels of length 34 is (21,18). If this same equation is applied to codes of length 17, two of the possible solutions are (11,10) and (11,7). The (11,10) pair when interlaced will total 21 ones, and similarly the (11,7) pair when interlaced will total 18 ones. Whereas the kernel of length 10 with the quad removed had a sequence quadruple form with codes of length 4, the 34 code with the quad removed would have a sequence quadruple form with codes of length 16. Appendix II gives a list of all possible codes of length 16 with weight 10 in the A code and weight 6 in the B code. This list does not include the results of the time reversal transformations which have to be included since weight is invariant under T_1 , T_2 and T . It was observed that if these codes of length 16 were interlaced to form codes of length 32 and then separated in the middle and the quad A=11, B=10 inserted, all conditions as far as the number of ones required in both interlace codes of 17 to

form codes of length 34 would be satisfied. An example might serve to clarify the preceding statements.

C=0101111111000110

D=1001001100001010

is a complementary pair of length 16. Using the interlace method to form a pair of length 32 gives

A=0110001110101111010000001101100

B=00110110111110101111010100111001 .

Splitting this A,B pair in the center and adding the quad to this center gives

A¹=0110001110101111 11 1010000001101100

B¹=0011011011111010 10 1111010100111001 .

Breaking this pair into standard (I II III IV) form gives

I=0101111111000110

II=01010000111001001

III=0101111111000110

IV=10101111000110110 .

I has 11 ones. Interlacing I with II, which has 7 ones, yields a total of 18 ones. Similarly III has 11 ones, and IV has 10 ones. When these are interlaced the code of length 34 has 21 ones. In the actual computer run the codes of length 16 with 6 ones were converted, by complementing, into codes with 10 ones. Therefore both the A and B codes contained 10 ones. An exhaustive list of these and their time reversals was generated. Each code of length 16 was then split in the center and a zero inserted for the ninth bit, forming an over all code of length 17 with 10 ones. A second list was then formed, using a one rather than a zero for the inserted ninth bit, forming codes of length 17 with 11 ones.

These two lists of numbers were then fed into the program of Figure 7.1 at block two, and the regular supplementary check program

was used. The codes with 11 ones were used for (I, III) and the codes with 10 ones were used for (II, $\overline{\text{IV}}$). When these codes were combined in the interlace scheme, I interlaced with II gave 21 ones and III interlaced with $\overline{\text{IV}}$ gave 18 ones. All four interlace combinations (I II III $\overline{\text{IV}}$), (II I III $\overline{\text{IV}}$), (I II $\overline{\text{IV}}$ III) and (II I $\overline{\text{IV}}$ III) were run exhaustively. The run time was approximately three hours and no codes were found. These partial searches for $n=34$ are documented here because they represent two fairly obvious approaches to the problem, and it would be very wasteful of time for someone to duplicate this effort. The program as given in the appendix can with a slight modification be used for an exhaustive search for kernels of length 34 or for a partial search based upon other "judicious guesses".

CHAPTER IX

CONCLUSIONS

The study which has been presented here had two principal objectives. The first objective was to make an exhaustive search for kernels of length 26 to determine whether they existed, and if they did exist, to determine the number of such kernels. The second objective was to develop a methodical scheme of code decomposition from composite codes back to their generating kernels. We shall summarize in this chapter the ways in which each of the major portions of this dissertation is related to the accomplishment of these objectives. Also a number of suggestions for further research will be presented.

The operations group and its family of theorems were originated to formalize the elimination of possible redundancies in the search for new kernels. The predetermination of the number of ones in each of the sequence quadruple vectors was a vital screen that eliminated many such redundancies. Also, the starting of both the I and III vectors from the same possible supplementary code groups eliminated an additional half of the possible codes. The theorems in the operations group chapter also were useful in the code decomposition part of the problem, since they show the necessary and sufficient conditions for the (I, II) and (III, IV) pairs to be complementary. Theorem 3.4, which states that if the sequence quadruple form is $(I \ II \ I \ \overline{II})$ then (I, II) must be a complementary pair, is most helpful since the process of checking for complementarity is very tedious by hand methods.

Hamming distances were the next major topic considered. They were found to be useful in a number of the proofs in the Hamming vector chapter. They were also a great aid in checking hand decomposition, since it was easy to determine whether $D(I, II) = D(III, IV) = n/4$ for composite codes or whether they had the characteristic distance in the case of kernels.

Hamming vectors were the most powerful tool developed in this paper for code decomposition. The theorems cover all known possible decompositions and also give strong support to the conjecture that kernels of length 50 do not exist. The Hamming vector concept should allow a fuller understanding of complementary sequences in general, due to its characteristic qualities such as anti-symmetry, and also because there are only four possible configurations of Hamming vectors for any code pair. Similarly in sequence quadruple form it is easy to check whether a decomposed code may be complementary by using its Hamming vector.

Several exhaustive searches for codes of various lengths were conducted during this research. The most important of these was the search for kernels of length 26. This search revealed that only one kernel existed. An exhaustive search for codes of length 16 and 20 revealed that all codes of these lengths were generated from shorter codes by standard methods. This gives considerable strength to the conjecture that these generating methods are the only ones.

The possibilities for future research in this field of complementary sequences are many. Some of these follow directly from this paper while others lie quite far afield from the ideas exposed here. An interesting extension of the present work would be an exhaustive search for kernels of length 34. It is felt that the cyclic complementary screen which was used in the partial search is not powerful enough, and a better screen is needed before this search can be conducted on an exhaustive basis using today's computers. The writer had considered the combination of both the cyclic and supplementary properties as screens in the same search. However, a full study was not made of the gain which might be obtained by this screening procedure. The programming

of such a screen would not prove too difficult and, since the cyclic property would logically be the first screen of the two to be used, the areas of search could still be catalogued quite easily.

If it would be possible to determine, before the search, the characteristic Hamming distances $D(I, II)$ and $D(II, IV)$ of kernels, these would act as a most potent screen in the search for new kernels. It is therefore felt that further research on characteristic Hamming distances of kernels could prove quite fruitful for future applications.

It was conjectured that kernels of length 50, 130 and 338 do not exist, based on several pieces of evidence, which although strong are not conclusive. Further investigation along these lines might offer a conclusive proof without the necessity for an exhaustive search, since an exhaustive search for these length codes approaches the impossible. During the investigation of kernels of length 50 the writer noticed an oddity which might be worthy of further investigation.

One kernel of length 10 is

$A=1001010001$

$B=1000000110.$

Divide both the A and B codes into segments of length 2, then

$A=10 \quad 01 \quad 01 \quad 00 \quad 01$

$B=10 \quad 00 \quad 00 \quad 01 \quad 10.$

Now let the symbols $10=A$, $01=\bar{A}$, $11=B$, and $00=\bar{B}$. Forming a code of length 50 by this method of using symbols of portions of the quad to represent kernels of length 10 gives

$K_1=A\bar{A}A\bar{B}\bar{A}$

$K_2=A\bar{B}\bar{B}A\bar{A}.$

This code pair of length 50 satisfies the parity check, and also equation 2.5 for the number of ones in both full codes and in the four half codes. Several examples were checked in this fashion for both kernels

of length 10, and all satisfied the same conditions as did this example, but of course none of them satisfied the necessary condition for complementarity. Along with helping to search for codes of length 50, this offers a possibility for research in that all kernels seem to have various characteristics in common. However, the quad seems more flexible. Of the four known kernels, the quad is the only one thus far discovered that can be used to build up composite codes by one method of generation and then these codes can be decomposed by a different method. Therefore a study of kernel characteristics might lead to easier ways of generating new kernels, or at least of determining whether they exist for the various possible lengths.

Another possible field of research is cyclic complementary codes. A communications system is more likely to use a cyclic complementary carrier modulation instead of straight complementary modulation due to the continuous nature of the carrier. Therefore an investigation of these codes seems quite in order.

Two other topics worthy of research in the complementary sequence field are the various correlation functions of complementary sequences and the frequency spectra of these sequences. For secure communications purposes it is important to have a code that is noise-like. A uniform distribution of ones and zeros gives a noise-like appearance to the code. For example, the complementary pair

A=11010001

B=11011110

is a typical complementary code pair of length 8. If the bits of the A code are considered in a cyclic fashion one, two, and three bits at a time, the results are remarkably uniform as is seen in Table 9.1. This uniformity in output would make it extremely difficult for someone

monitoring the code to decide if he had noise or an actual signal on his receiver. However, it is seen that the B code is not nearly so uniform and would be easier to detect as a signal.

A code

Bits	Number	Two Bits	Number	Three Bits	Number
1	4	11	2	111	1
0	4	10	2	110	1
		01	2	101	1
		00	2	011	1
				100	1
				010	1
				001	1
				000	1

B code

1	6	11	4	111	2
0	2	10	2	110	2
		01	2	101	2
		00	0	011	2
				100	0
				010	0
				001	0
				000	0

Noise uniformity of a complementary pair of length 8.

TABLE 9.1

An investigation of a large number of codes by the above technique would be useful in determining the most noise-like code pairs, where noise-like is defined as a uniform output with bits taken one, two, three, etc. at a time. This might in turn lead to some general characteristic of most-noise-like complementary sequence pairs.

Another important study would be the autocorrelation function of each code pair. This is synonymous with the output of a code detected by its matched filter. For example, the autocorrelation of the code A given above is

1 0 -3 0 -1 0 -1 8 -1 0 -1 0 -3 0 1

where the outputs in time are read from left to right. The autocorrelation of the B code is

-1 0 3 0 1 0 1 8 1 0 1 0 3 0 -1.

The autocorrelation functions are important from two standpoints. First, if a portion of the system should need to be shut down for repairs or routine maintenance, a good autocorrelation function would allow operation on just one carrier. Second, the frequency spectrum of the transmitted signal is the Fourier transform of this autocorrelation function. Therefore, one way to study the spectrum is first to autocorrelate and then to take the Fourier transform of the transmitted signals. The more widespread the spectrum, the more difficult is a jamming procedure.

A study of the crosscorrelation functions of complementary sequence pairs is also very important because of the difficulty of isolating the A code RF carrier from the B code receiver. It was pointed out in chapter 4 that because the A code and the B code were orthogonal, at exact match cross talk would be no problem. However, when the signals are not at exact match cross talk might be a large source of noise. An investigation of the crosscorrelation functions of the A and B code pairs might lead to a class of complementary sequences where the problem of crosstalk is minimized.

Another class of correlation functions which are important are those where cyclic errors are created in the received signal detected by the matched filter. These errors could be caused by a linear shift in

the phase of the received signal. There may be certain classes of complementary sequences where the loss in coding signal-to-noise ratio is minimized when noise is caused by an error of this type. This particular class of signals would be extremely important in an air search radar application.

A study of the maximum number of different composite codes for any length would be an important contribution to the field, because for anti-jam reasons it is very important to be able to change codes, and the more codes with good characteristics that are available the better.

The last study to be recommended is an analysis of the autocorrelation functions, frequency spectra, cross correlation functions, and linear phase shift correlation functions for codes of lengths from 100 to 200. The purpose of this investigation would be to determine if certain kernels are capable of forming composite codes of better characteristics than other kernels for radar applications.

In conclusion, the writer feels that complementary sequence pairs will prove very important in future applications to both radar and communications schemes. However, much research must be done to pick optimal classes of these codes for such applications. This paper has offered many tools to be used by future investigators of complementary sequences. These tools have found important applications in their present state, but additional investigations along the lines of operations groups, Hamming weights and Hamming vectors should further increase their usefulness.

GLOSSARY OF SYMBOLS AND TERMS

(A, B)	A complementary pair of binary sequences, see equation 2.2.
$(I \ II \ III \ IV)$	A complementary pair of binary sequences in sequence quadruple form, see equation 3.1.
$D(U, V)$	The Hamming distance of two binary vectors, see equation 4.1.
$H(U, V)$	The Hamming vector of two binary vectors, see the first page of Chapter 5.
*	Means interlace two Hamming vectors in a prescribed manner, see Theorem 5.11, step 2.
\rightarrow	Is transformed into.
<u>A</u>	The time inverse of a binary sequence or vector A.
Alter	Means to complement every other bit of both codes of a complementary code pair.
Complementary code	A pair of binary sequences which satisfy equation 2.2, also called a complementary sequence pair, a complementary sequence and sometime just a pair or code.
Complement a code	Means change the sign or take the complement of a binary sequence.
Composite code	A complementary code which is reducible to a shorter code length by standard methods.
Cyclic complementary code	A pair of binary sequence which satisfy equation 6.3.
Hamming Distance	The number of bits in which two binary sequence differ, notation is $D(A, B)$, see equation 4.1.
Hamming vector	A binary vector or sequence formed by the modulo 2 sum of binary sequences.
Hamming weight	Hamming distance of a binary sequence or vector from the null vector. It is the number of ones in a binary vector.
Kernel	A set of complementary codes which is irreducible. The term is sometimes used for a member of the set.
Quad	A kernel of length 2.
Sequence Quadruple	A complementary code pair expressed as in equation 3.1.
Supplementary code	A quadruple of binary sequences which satisfy equation 6.1.
Time inverse	Means to make the first bit of a binary sequence the last bit, the second bit next to last, and so on; also called time reverse.
Weight	See Hamming weight.

BIBLIOGRAPHY

1. M. J.E. Golay, "Multi-Slit Spectrometry",
J. Opt. Soc. Am., Vol. 39, pp. 437-444, June 1949.
2. M. J.E. Golay, "Static Multi-Slit Spectrometry and Its
Application to the Panoramic Display of Infrared Spectra",
J. Opt. Soc. Am., Vol. 41, pp. 468-472, July 1951.
3. J. I. Marcum, "A Statistical Theory of Target Detection by
Pulsed Radar", RAND Research Memo. RM-754,
December 1947, also IRE Trans. Info. Theory, Vol. IT-6
Number 2, April 1960.
4. J. R. Klauder, A. C. Price, S. Darlington and W. J. Albersheim,
"The Theory and Design of Chirp Radars", B. S. T. J.,
Vol. 39, pp. 745-809, July 1960.
5. L. Baumert, M. Esterling, S. W. Golomb, A. Viterbi,
"Coding Theory and Its Applications to Communications
Systems", Jet Prop. Lab. Tech. Report 32-67, March 1961.
6. R. C. Tittsworth and L. R. Welch, "Modulation by Random and
Pseudo-Random Sequences", Jet Prop. Lab. Prog. Rpt.
20-387, June 1959.
7. R. C. Tittsworth, "Correlation Properties of Random-Like
Periodic Sequences", Jet Prop. Lab. Prog. Rpt. 20-391,
October 1959.
8. B. Elspas, "A Radar System Based on Statistical Estimation
and Resolution Considerations", App. Elect. Lab.
Stanford Univ. Tech. Rpt. 361-1, August 1955.
9. M. J.E. Golay, "Complementary Series", IRE Trans. Info.
Theory, Vol. IT-7, pp. 82-87, April 1961.
10. R. W. Hamming, "Error Detecting and Error Correcting Codes",
B. S. T. J., Vol. 29, pp. 147-160, April 1950.
11. G. R. Welty, "Quaternary Codes for Pulsed Radar", IRE Trans.
Info. Theory, Vol. IT-6, pp. 400-408, June 1960.
12. C. W. Erickson, "Clutter Cancelling in Autocorrelation
Functions by Binary Sequence Pairing", USNEL
Rpt. 1047, June 1961.
13. J. B. Kruskal, "Golay's Complementary Series", IRE Trans.
Info. Theory, Vol. IT-7, pp. 273-276, October 1961.
14. M. J.E. Golay, personal correspondence dated 4 Sept. 61.
15. G. Birkhoff, S. Mac Lane, "A Survey of Modern Algebra",
MacMillan Co., 8th printing, 1959.

16. J.K. Senior, Diagrams showing inclusion relations, and generating relations, for groups of order 32 and factors. A copy of this unpublished work, done at the University of Chicago, dating from approximately 1934, is in possession of Randolph Church, Dept. of Mathematics and Mechanics, U.S. N. Postgraduate School.
17. J.K. Senior, M. Hall and P. Hall. Tables of generating relations and diagrams showing inclusion relations for each group of order 2^n , $n = 1, 2, \dots, 6$. This improved and extended version of the work cited above is under consideration for publication. The first author kindly made a copy available toward the end of the investigation reported in this dissertation.
18. G.A. Miller, "The Regular Substitution Groups Whose Orders Are Less Than 48", Quarterly Journal of Math. Vol. 28, pp. 232-284, 1896. Also "The Collected Works of George Abram Miller", Vol. 1, Univ. of Ill. Press, 1935.
19. M.J.E. Golay, personal correspondence dated 17 Oct 61.
20. M.J.E. Golay, personal correspondence dated 6 Oct. 61.
21. M.J.E. Golay, "Note on Complementary Series", Proc IRE, Vol. 50, pp. 84, January 1962.
22. A. Blanc-Lapierre and R. Fortet, "Theorie des Fonctions Aleatoires", Masson et cie, 1953.
23. G. L. Turin, "An Introduction to Matched Filters", IRE Trans. Info. Theory, Vol. IT-6, pp. 311-329, June 1960.

EXPLANATION OF FORMAT OF THE APPENDICIES

Appendix I on matched filters is self-explanatory. Appendix II is the result of an exhaustive search for codes of length 16. There were 96 code pairs found by the computer after the program screens had eliminated most of the redundancies. The format of Appendix II gives the A code, the B code, $H(A,B)$, and $H[T_1(A,B)]$ which signifies time reversing the A code. It is worthwhile to note that each code pair and the pair formed by the time reverse of A were decomposable. From these 192 possible decompositions 48 were interlace, 48 were time sequence, and 96 were 2^r special method, with 48 each of two different kinds.

Appendix III is the result of an exhaustive search for codes of length 20. The format of Appendix III gives the A code, the B code, if it is required, the time reversal operation used to put $H(A,B)$ in standard form, $H(A,B)$ and from which of the 2 kernels of length 10 the generating codes came. All of the 24 code pairs or their time inverse were decomposable. There are 12 from each kernel, 8 of each of these interlace and the other 4 are time sequence.

Appendix IV is a diagram of all the subgroups of the operations group. It starts with the subgroup of order 1 and works up to those of order 16. An x in the row indicates that the operation at the top of the column is a member of the subgroup.

Appendix V is the CDC 1604 computer program in AR format and also in machine language for the 26 length search. There are slight modifications in the various sections of the program required for other length codes. Appendix VI is also in both AR and machine language, and is for codes of length 34. This program parallels Figure 8.1, whereas the program in Appendix V parallels Fig. 7.1.

APPENDIX I

MATCHED FILTERS

A Matched Filter is by definition a filter which maximizes the peak signal-to-noise ratio. Deterioration of the signal wave form is accepted in order to obtain the desired maximum ratio.

One tool needed for the derivation of the matched filter characteristics is the Schwarz inequality which is a special case of the Holder inequality.²² One representation of the Schwarz inequality is

$$\left| \int_{-\infty}^{\infty} x(w) y(w) dw \right| < \sqrt{\int_{-\infty}^{\infty} |x(w)|^2 dw \int_{-\infty}^{\infty} |y(w)|^2 dw} \quad (\text{A } 1.1)$$

It is to be noted that the equality holds when $y(w)$ is the complex conjugate of $x(w)$. Rewriting equation A 1.1 gives

$$\frac{\left| \int_{-\infty}^{\infty} x(w) y(w) dw \right|^2}{\int_{-\infty}^{\infty} |x(w)|^2 dw \int_{-\infty}^{\infty} |y(w)|^2 dw} < 1 \quad (\text{A } 1.2)$$

The derivation for the matched filter given here closely parallels that of Turin.²³

Let $f(t)$ be a signal impressed across a filter whose frequency characteristic is $H(jw)$. $F(jw)$, the frequency spectrum of $f(t)$, is given by the Fourier transform of $f(t)$.

$$F(jw) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt.$$

$G(jw)$ is the frequency spectrum of the output after passing through the filter

$$G(jw) = H(jw) F(jw),$$

or transforming back to the time domain the output signal voltage is

$$g(t) = \int_{-\infty}^{\infty} H(j\omega) F(j\omega) e^{j\omega t} d\omega. \quad (\text{A } 1.3)$$

Since at some time, $t = \Delta$, $g(t)$ must be a maximum,

$$\max g(t) = g(\Delta) = \int_{-\infty}^{\infty} H(j\omega) F(j\omega) e^{j\omega \Delta} d\omega. \quad (\text{A } 1.4)$$

To complete the derivation it will be necessary to obtain the total noise power, and the total power in the signal. If the noise is assumed to be white noise of N watts per cycle, the output noise power density is

$$N_o = N |H(j\omega)|^2,$$

and the output noise power is therefore

$$\int_{-\infty}^{\infty} N_o d\omega = \int_{-\infty}^{\infty} N |H(j\omega)|^2 d\omega. \quad (\text{A } 1.5)$$

The total energy contained in the signal is

$$E = \int_{-\infty}^{\infty} f^2(t) dt,$$

which is also

$$E = \int_{-\infty}^{\infty} |F(j\omega)|^2 d\omega, \quad (\text{A } 1.6)$$

by Plancherel's Theorem. It is to be noted that E is a constant not dependent upon the filter used.

The output power signal-to-noise ratio for the maximum output is therefore

$$\frac{|g(\Delta)|^2}{N_0} = \frac{\left| \int_{-\infty}^{\infty} F(j\omega) H(j\omega) e^{j\omega\Delta} d\omega \right|^2}{N \int_{-\infty}^{\infty} |H(j\omega)|^2 d\omega}.$$

Dividing this equation by a constant will not effect the time of maximum signal output; therefore dividing by E , equation A 1.6 , gives

$$\frac{N |g(\Delta)|^2}{E N_0} = \frac{\left| \int_{-\infty}^{\infty} F(j\omega) H(j\omega) e^{j\omega\Delta} d\omega \right|^2}{\int_{-\infty}^{\infty} |H(j\omega)|^2 d\omega \int_{-\infty}^{\infty} |F(j\omega)|^2 d\omega} \quad (\text{A 1.7})$$

If the right hand side of equation (A 1.7) is compared to the Schwarz inequality (A 1.2) with $x(\omega) = F(j\omega) e^{j\omega\Delta}$ $y(\omega) = H(j\omega)$, the two expressions are the same. But for the expression to be maximum $y(\omega)$ must equal the complex conjugate of $x(\omega)$, therefore $H(j\omega) = F(-j\omega) e^{-j\omega\Delta}$.

A matched filter is therefore a filter whose frequency characteristic is the complex conjugate of the signal spectrum to which is matched.

The output signal for a given signal applied to a matched filter is therefore

$$g(t) = \int_{-\infty}^{\infty} |F(j\omega)|^2 e^{j\omega(t-\Delta)} d\omega,$$

at $t = \Delta$ the output is

$$g(\Delta) = \int_{-\infty}^{\infty} |F(j\omega)|^2 d\omega.$$

This is the same as equation A 1.6 which is all the energy contained in the signal. Therefore at exact match all the energy in the pulse is in the output signal.

APPENDIX II

Exhaustive Search for Codes of Length 16.

	A	B	H	$[T_1(A, B)]$
1.	1101000111011110	1000010010001011	$(01)^8$	$H=1^8 0^8$
2.	1001010111001111	1100000010010101	$(01)^8$	$(0^2 1^2)^4$
3.	1111001101010110	1010011000000011	$(01)^8$	$(1^2 0^2)^4$
4.	1011011101000111	1110001000010010	$(01)^8$	$0^8 1^8$
5.	0101100111111100	0000110010101001	$(01)^8$	$(0^2 1^2)^4$
6.	0001110111101101	0100100010111000	$(01)^8$	$1^8 0^8$
7.	0011111101100101	0110101000110000	$(01)^8$	$(1^2 0^2)^4$
8.	01111101101110100	0010111000100001	$(01)^8$	$0^8 1^8$
9.	1101011100011011	0010100000011011	$1^8 0^8$	$(1^4 0^4)^2$
10.	1100011001011111	0000101010010011	$(1^2 0^2)^4$	$(1^4 0^4)^2$
11.	1110010011010111	1110010000101000	$0^8 1^8$	$(0^4 1^4)^2$
12.	1111010110010011	1100011010100000	$(0^2 1^2)^4$	$(0^4 1^4)^2$
13.	0101111100111001	0110110000001010	$(0^2 1^2)^4$	$(1^4 0^4)^2$
14.	0100111001111101	0100111010000010	$0^8 1^8$	$(1^4 0^4)^2$
15.	0110110011110101	1010000000111001	$(1^2 0^2)^4$	$(0^4 1^4)^2$
16.	0111110110110001	1000001010110001	$1^8 0^8$	$(0^4 1^4)^2$
17.	0101110001101111	0000100100111010	$(01)^8$	$1^8 0^8$
18.	0101100100111111	0000110001101010	$(01)^8$	$(1^4 0^4)^2$
19.	0101011011001111	0000001110011010	$(01)^8$	$(1^4 0^4)^2$
20.	0101001110011111	0000011011001010	$(01)^8$	$1^8 0^8$
21.	1111110001100101	1010100100110000	$(01)^8$	$(0^4 1^4)^2$
22.	1111100100110101	1010110001100000	$(01)^8$	$0^8 1^8$
23.	1111011011000101	1010001110010000	$(01)^8$	$0^8 1^8$
24.	1111001110010101	1010011011000000	$(01)^8$	$(0^4 1^4)^2$

	A	B	H	$[T_1(A, B)]$
25.	0111010001111011	0010000100101110	$(01)^8$	1^8_0
26.	0110010100111111	0011000001101010	$(01)^8$	$(1^2_0)^2_4$
27.	0101011011110011	0000001110100110	$(01)^8$	$(1^2_0)^2_4$
28.	0100011110110111	0001001011100010	$(01)^8$	1^8_0
29.	1111110001011001	1010100100001100	$(01)^8$	$(0^2_1)^2_4$
30.	1110110100011101	1011100001001000	$(01)^8$	0^8_1
31.	1101111011010001	1000101110000100	$(01)^8$	0^8_1
32.	110011110010101	1001101011000000	$(01)^8$	$(0^2_1)^2_4$
33.	0111110101110010	1000001001110010	1^8_0	$(0^2_1)^2_4$
34.	0010110101110111	0010001001111000	$(0^4_1)^4_2$	$(1^2_0)^2_4$
35.	0111011111010010	1000011100100010	$(1^4_0)^4_2$	$(1^2_0)^2_4$
36.	0010011111010111	0010011100101000	0^8_1	$(1^2_0)^2_4$
37.	1000110101111101	1000110110000010	0^8_1	$(0^2_1)^2_4$
38.	1101110101111000	0010110110001000	$(1^4_0)^4_2$	$(0^2_1)^2_4$
39.	1000011111011101	1000100011010010	$(0^4_1)^4_2$	$(0^2_1)^2_4$
40.	1101011111011000	0010100011011000	1^8_0	$(0^2_1)^2_4$
41.	1101110100011110	1000100001001011	$(01)^8$	$(1^4_0)^2_4$
42.	1001110001011111	1100100100001010	$(01)^8$	$(0^2_1)^2_4$
43.	1111010100110110	1010000001100011	$(01)^8$	$(1^2_0)^2_4$
44.	1011010001110111	1110000100100010	$(01)^8$	$(0^4_1)^2_4$
45.	0101111110011100	0000101011001001	$(01)^8$	$(0^2_1)^2_4$
46.	0001111011011101	0100101110001000	$(01)^8$	$(1^4_0)^2_4$
47.	0111011110110100	0010001011100001	$(01)^8$	$(0^4_1)^2_4$
48.	0011011011110101	0110001110100000	$(01)^8$	$(1^2_0)^2_4$

	A	B	H	$[T_1(A, B)]$
49.	1101000101111011	0010000110001011	$(1^4 0^4)^2$	$1^8 0^8$
50.	1100010101101111	0000100110100011	$(1^2 0^2)^4$	$1^8 0^8$
51.	1110110101000111	1110001001001000	$(0^4 1^4)^2$	$0^8 1^8$
52.	1111100101010011	1100101001100000	$(0^2 1^2)^4$	$0^8 1^8$
53.	0101001111111001	0110000011001010	$(0^2 1^2)^4$	$1^8 0^8$
54.	0100011111101101	0100100011100010	$(0^4 1^4)^2$	$1^8 0^8$
55.	0110111111000101	1010001100001001	$(1^2 0^2)^4$	$0^8 1^8$
56.	0111101111010001	1000010000101110	$(1^4 0^4)^2$	$0^8 1^8$
57.	0111010011011110	1000010000101110	$(1^4 0^4)^2$	$1^8 0^8$
58.	0011010110011111	0000011010101100	$(0^2 1^2)^4$	$1^8 0^8$
59.	0101110011110110	1001000000111010	$(1^2 0^2)^4$	$1^8 0^8$
60.	0001110110110111	0001001010111000	$(0^4 1^4)^2$	$1^8 0^8$
61.	1011011100011101	1011100000010010	$(0^4 1^4)^2$	$0^8 1^8$
62.	1111011001011100	0011101010010000	$(1^2 0^2)^4$	$0^8 1^8$
63.	1001111100110101	1010110000000110	$(0^2 1^2)^4$	$0^8 1^8$
64.	1101111001110100	0010111010000100	$(1^4 0^4)^2$	$0^8 1^8$
65.	0111011101001011	0010001000011110	$(01)^8$	$(1^4 0^4)^2$
66.	0110001101011111	0011011000001010	$(01)^8$	$(1^2 0^2)^4$
67.	0101111101100011	0000101000110110	$(01)^8$	$(1^2 0^2)^4$
68.	0100101101110111	0001111000100010	$(01)^8$	$(1^4 0^4)^2$
69.	1111010111001001	1010000010011100	$(01)^8$	$(0^2 1^2)^4$
70.	1110000111011101	1011010010001000	$(01)^8$	$(0^4 1^4)^2$
71.	1101001011100001	1000100010110100	$(01)^8$	$(0^4 1^4)^2$
72.	1100100111110101	1001110010100000	$(01)^8$	$(0^2 1^2)^4$

	A	B	H	$[T_1(A, B)]$
73.	1101011100100111	0010100000100111	1^8_0	$(0^2_1 2^4)$
74.	1101001001110111	0010001010000111	$(1^4_0 4^2)$	$(1^2_0 2^4)$
75.	1101100011010111	1101100000101000	0^8_1	$(0^2_1 2^4)$
76.	1101110110000111	1101001010001000	$(0^4_1 4^2)$	$(0^2_1 2^4)$
77.	0111011100101101	0111100000100010	$(0^4_1 4^2)$	$(1^2_0 2^4)$
78.	0111001001111101	0111001010000010	0^8_1	$(1^2_0 2^4)$
79.	0111100011011101	1000100000101101	$(1^4_0 4^2)$	$(0^2_1 2^4)$
80.	0111110110001101	1000001010001101	1^8_0	$(0^2_1 2^4)$
81.	1100010111110110	1001000010100011	$(01)^8$	1^8_0
82.	1001010111110011	1100000010100110	$(01)^8$	$(0^4_1 4^2)$
83.	1100111101010110	1001101000000011	$(01)^8$	$(1^4_0 4^2)$
84.	1001111101010011	1100101000000110	$(01)^8$	0^8_1
85.	0110010111111100	0011000010101001	$(01)^8$	$(0^4_1 4^2)$
86.	0011010111111001	0110000010101100	$(01)^8$	1^8_0
87.	0110111101011100	0011101000001001	$(01)^8$	0^8_1
88.	0011111101011001	0110101000001100	$(01)^8$	$(1^4_0 4^2)$
89.	0111110101001110	1000001001001110	1^8_0	$(1^4_0 4^2)$
90.	0011100101011111	0000101001101100	$(0^2_1 2^4)$	$(1^4_0 4^2)$
91.	0101111111000110	1001001100001010	$(1^2_0 2^4)$	$(1^4_0 4^2)$
92.	0001101111010111	0001101100101000	0^8_1	$(1^4_0 4^2)$
93.	1011000101111101	1011000110000010	0^8_1	$(0^4_1 4^2)$
94.	1111010101101100	0011100101010000	$(1^2_0 2^4)$	$(0^4_1 4^2)$
95.	1001001111110101	1010000011000110	$(0^2_1 2^4)$	$(0^4_1 4^2)$
96.	1101011111100100	0010100011100100	1^8_0	$(0^4_1 4^2)$

APPENDIX III Exhaustive Search for Codes of Length 20.

	A	B	OP	H	Kernel
1.	01010000110000100110	01100100000011110101	T ₁	0 ¹⁰ ₁ 10	2
2.	00110101000010000011	11000001001101010011	T ₁	0 ¹⁰ ₁ 10	1
3.	10010100011000000110	01100000010111010110	T ₁	0 ¹⁰ ₁ 10	1
4.	00110001010110100000	00000101100101110011	T ₁	0 ¹⁰ ₁ 10	2
5.	01000100001001111000	00010001011100101101	I	(01) ¹⁰	2
6.	00000101001000111001	01010000011101101100	I	(01) ¹⁰	2
7.	01010000100001101100	00000101110100111001	I	(01) ¹⁰	2
8.	00010001100000101101	01000100110101111000	I	(01) ¹⁰	2
9.	011111000001001000100	00101101011100010001	I	(01) ¹⁰	2
10.	00111001001000000101	01101100011101010000	I	(01) ¹⁰	2
11.	01101100100001010000	00111001110100000101	I	(01) ¹⁰	2
12.	00101101100000010001	01111000110101000100	I	(01) ¹⁰	2
13.	01101000000011000101	010111001100000010110	T ₂	0 ¹⁰ ₁ 10	2
14.	00001101000100011010	01011000010001001111	I	(01) ¹⁰	1
15.	00001100010001011010	010110010001000001111	I	(01) ¹⁰	1
16.	10100101000100110000	11110000010001100101	I	(01) ¹⁰	1
17.	10100100010001110000	11110001000100100101	I	(01) ¹⁰	1
18.	10000001101001010001	01110101100110000001	T ₂	0 ¹⁰ ₁ 10	1
19.	01101001000100000011	00111100010001010110	I	(01) ¹⁰	1
20.	01101000010001000011	00111101000100010110	I	(01) ¹⁰	1
21.	11000001000100101001	10010100010001111100	I	(01) ¹⁰	1
22.	11000000010001101001	10010101000100111100	I	(01) ¹⁰	1
23.	00100000110011010100	11010100111100000100	T ₂	0 ¹⁰ ₁ 10	1
24.	00001001100101000011	00111101010110010000	T ₂	0 ¹⁰ ₁ 10	2

Subgroups of the Operations Group

130

	I	C ₁	C ₂	C	T ₁	T ₂	T	O	N	M	L	K	J	H	G	∅	A ₁	A ₂	F	D	Z	Y	X	W	V	U	S	R	Q̃	P	B	τ
	4	x	x			x				x																						
	4	x	x				x									x																
	4	x	x						x		x																					
	4	x	x									x		x																		
	4	x	x										x																			
	4	x		x												x																
	4	x		x					x																							
	4	x	x		x																											
	4	x	x			x								x																		
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												
	4	x		x																												

[illegible]

	I	C ₁	C ₂	C	T ₁	T ₂	T	O	N	M	L	K	J	H	G	∅	A ₁	A ₂	F	D	Z	Y	X	W	V	U	S	R	Q	P	B	π
8	x	x	x	x	x			x	x		x																					
8	x	x	x	x		x				x		x		x																		
8	x	x	x	x			x						x		x	x																
8	x	x			x	x	x	x		x					x																	
8	x	x	x	x	x			x				x	x	x		x																
8	x	x				x			x	x	x		x																			
8	x		x				x		x		x	x		x	x																	
8	x			x	x	x	x		x					x		x																
8	x							x	x		x	x																				
8	x												x	x																		
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x																															
8	x	</																														

	I	C ₁	C ₂	C	T ₁	T ₂	T	O	N	M	L	K	J	H	G	∅	A ₁	A ₂	F	D	Z	Y	X	W	V	U	S	R	Q	P	B	π	
8	x			x			x						x				x												x				
8	x		x				x						x					x		x											x		x
8	x	x						x			x							x		x	x												
8	x	x						x	x		x						x		x			x											
8	x		x						x			x						x							x								
8	x		x						x			x					x		x														
8	x			x						x									x	x													
8	x									x							x		x														
8	x																x																
8	x			x													x		x														
8	x																x																
8	x																x																
16	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x																	
16	x	x	x	x	x			x	x		x						x	x	x	x	x	x	x	x									
16	x	x	x	x	x			x	x		x						x	x															
16	x	x	x	x				x		x							x	x	x														
16	x	x	x	x					x								x	x	x														
16	x	x	x	x					x								x	x	x														
16	x	x	x	x						x							x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x													x	x	x														
16	x	x	x	x		</																											

APPENDIX V

REMARKS ON COMPUTER PROGRAM

In Appendix V the addresses from 60000 to 60035 are block 1 of Figure 7.1. All numbers used in the program are octal. The program as written is for codes of length 26, so most of the program deals with half length codes of length 13. Words to be changed for different length codes are 60000 (mask of ones for half length code as determined by weight), 60001 (length of half code minus one [upper half], and length of memory word minus length of code [lower half]), 60004 (length of half code), 60034 (last sequence in code list). For exactly the same reasons respectively change 60015, 60016, 60021 and 60032.

Addresses 17000 to 17045 separate the sequences into blocks by supplementary number. Index register 2 must be entered with 100 before starting. Addresses to be changed here are 17000 (mask of first 3 and last 3 bits), 17005 (contains the total number of sequences for both half length codes), 17007 (contains the number of sequences with 7 ones), 17015 (contains a mask for half length codes), 17026 (contains the number of sequences with 9 ones), 17032, 17033 (have for their instruction address the base word 50000 plus the number of half codes with 7 ones), 17034 (contains a mask for code length).

Addresses 10000 to 10106 are the repetitive part of the program. The only memory address needing modification here for different length codes is 10106 which contains SVN a mask equal to the one half code length.

Addresses 7200 to 7252 contain the unpacking subroutine and the like-unlike subroutine. 7201, 7207, 7215, and 7223 all contain the half code length. Similarly 7204, 7207, 7220, 7223 and 7230 all contain the full code length or the length modified by one or two to fit the program needs.

The 15000 to 15100 series is all possible bit arrangements for the first 3 and the last 3 bits of codes of length 13, while 16000 to 16100 is a list of supplementary block numbers in one to one correspondence with this list.

The 10300 and 10400 series respectively contain the number of codes in each supplementary block for the half length codes of weight 9 and weight 7.

16000 to 16020 is the final list of the 18 supplementary block numbers and is in one to one correspondence with the 10300 and 10400 series.

				ORG	60000
60000	10 0 00177			ENA	0 177
	20 0 60035			STA	0 KEEP
60001	50 1 00006	REP		ENI	1 6
	05 0 00043			ALS	0 43
60002	22 3 60006	IT		AJP	3 CHG
	50 0 00000			ENI	0 0
60003	07 0 00001	BACK		LLS	0 1
	04 0 00000			ENQ	0 0
60004	54 2 00015			ISK	2 15
	75 0 60002			SLJ	0 IT
60005	72 0 60035	AG		RAO	0 KEEP
	75 0 60001			SLJ	0 REP
60006	55 1 60003	CHG		IJP	1 BACK
	07 0 00001			LLS	0 1
60007	04 0 00000			ENQ	0 0
	22 0 60011			AJP	0 REC
60010	50 2 00000	WOF		ENI	2 0
	75 0 60005			SLJ	0 AG
60011	12 0 60035	REC		LDA	0 KEEP
	20 6 50000			STA	6 50000
60012	51 6 00001			INI	6 1
	50 0 00000			ENI	0 0
60013	64 0 60034			EQS	0 NTVS
	76 1 60010			SLS	1 WOF
60014	50 0 00000			ENI	0 0
	57 6 60300			SIL	6 60300
60015	10 0 00777			ENA	0 777
	20 0 60035			STA	0 KEEP
60016	50 1 00010	PER		ENI	1 10
	05 0 00043			ALS	0 43
60017	22 3 60023	TI		AJP	3 GHC
	50 0 00000			ENI	0 0
60020	07 0 00001	KCAB		LLS	0 1
	04 0 00000			ENQ	0 0
60021	54 2 00015			ISK	2 15
	75 0 60017			SLJ	0 TI
60022	72 0 60035	GA		RAO	0 KEEP
	75 0 60016			SLJ	0 PER
60023	55 1 60020	GHC		IJP	1 KCAB
	07 0 00001			LLS	0 1
60024	04 0 00000			ENQ	0 0
	22 0 60026			AJP	0 CER
60025	50 2 00000	FOW		ENI	2 0
	75 0 60022			SLJ	0 GA
60026	12 0 60035	CER		LDA	0 KEEP
	20 6 50000			STA	6 50000
60027	51 6 00001			INI	6 1
	50 0 00000			ENI	0 0
60030	64 0 60032			EQS	0 SVTN
	76 1 60025			SLS	1 FOW
60031	50 0 00000			ENI	0 0
	76 0 04321			SLS	0 4321
60032	00 0 00000	SVTN		OCT	17760
	00 0 17760				

60033	50 0 00000		ENI	0 0
	50 0 00000			
60034	00 0 00000	NTVS	OCT	17700
	00 0 17700			
60035	00 0 00000	KEEP	OCT	0
	00 0 00000			
			ORG	17000
17000	04 0 16007	START	ENQ	0 16007
	44 1 50000		LDL	1 50000
17001	64 2 15000		EQS	2 15000
	75 0 17045		SLJ	0 WHOA
17002	12 2 16000		LDA	2 16000
	05 0 00030		ALS	0 30
17003	70 1 50000		RAD	1 50000
	51 1 00001		INI	1 1
17004	50 2 00100		ENI	2 100
	50 0 00000		ENI	0 0
17005	54 3 04477		ISK	3 4577
	75 0 17000		SLJ	0 START
17006	50 1 00000		ENI	1 0
	50 2 00000		ENI	2 0
17007	50 5 03264	TOP	ENI	5 3264
	50 0 00000		ENI	0 0
17010	04 0 00777	TRY	ENQ	0 777
	50 0 00000		ENI	0 0
17011	06 0 00030		QLS	0 30
	12 1 15500		LDA	1 15500
17012	05 0 00030		ALS	0 30
	20 2 30000		STA	2 30000
17013	66 5 50000		MEQ	5 50000
	75 0 17020		SLJ	0 NEXT
17014	72 2 30001		RAO	2 30001
	12 5 50000		LDA	5 50000
17015	04 0 17777		ENQ	0 17777
	50 0 00000		ENI	0 0
17016	47 3 30002	TAR	STL	3 30002
	50 0 00000		ENI	0 0
17017	51 3 00001		INI	3 1
	75 0 17010		SLJ	0 TRY
17020	51 1 00001	NEXT	INI	1 1
	51 2 00400		INI	2 400
17021	10 0 00400		ENA	0 400
	05 0 00030		ALS	0 30
17022	50 0 00000		ENI	0 0
	70 0 17016		RAD	0 TAR
17023	50 3 00000		ENI	3 0
	50 0 00000		ENI	0 0
17024	54 6 00022		ISK	6 22
	75 0 17007		SLJ	0 TOP
17025	50 1 00000		ENI	1 0
	50 2 00000		ENI	2 0
17026	50 5 01313	POT	ENI	5 1313
	50 0 00000		ENI	0 0
17027	04 0 00777	YRT	ENQ	0 777
	50 0 00000		ENI	0 0

17030	06 0 00030		QLS	0 30
	12 1 15500		LDA	1 15500
17031	05 0 00030		ALS	0 30
	20 2 20000		STA	2 20000
17032	66 5 53264		MEQ	5 53264
	75 0 17037		SLJ	0 TXEN
17033	72 2 20001		RAO	2 20001
	12 5 53264		LDA	5 53264
17034	04 0 17777		ENQ	0 17777
	50 0 00000		ENI	0 0
17035	47 3 20002	RAT	STL	3 20002
	50 0 00000		ENI	0 0
17036	51 3 00001		INI	3 1
	75 0 17027		SLJ	0 YRT
17037	51 1 00001	TXEN	INI	1 1
	51 2 00200		INI	2 200
17040	10 0 00200		ENA	0 200
	05 0 00030		ALS	0 30
17041	50 0 00000		ENI	0 0
	70 0 17035		RAD	0 RAT
17042	50 3 00000		ENI	3 0
	50 0 00000		ENI	0 0
17043	54 6 00022		ISK	6 22
	75 0 17026		SLJ	0 POT
17044	76 0 00000		SLS	0 0
	50 0 00000		ENI	0 0
17045	50 6 77777	WHOA	ENI	6 77777
	76 0 04444		SLS	0 4444
			ORG	10000
10000	10 0 00246	PERO	ENA	0 246
	50 0 00000		ENI	0 0
10001	15 4 15500		SUB	4 15500
	15 1 15500		SUB	1 15500
10002	15 2 15500		SUB	2 15500
	50 3 00022		ENI	3 22
10003	64 3 15500		EQS	3 15500
	75 0 10005		SLJ	0 BK
10004	75 0 10010		SLJ	0 COMPR
	50 0 00000		ENI	0 0
10005	54 2 0002z	BK	ISK	2 21
	75 0 10000		SLJ	0 PERO
10006	54 1 00021	SKB	ISK	1 21
	75 0 10000		SLJ	0 PERO
10007	76 0 00000		SLS	0
	50 0 00000		ENI	0 0
10010	57 1 10074	COMPR	SIL	1 BAG
	12 0 10074		LDA	0 BAG
10011	24 0 10104		MUI	0 TWOH
	14 0 10062		ADD	0 COREA
10012	20 0 10033		STA	0 AMC
	56 4 10075		SIU	4 FAG
10013	12 0 10075		LDA	0 FAG
	24 0 10104		MUI	0 TWOH
10014	70 0 10033		RAD	0 AMC
	56 2 10076		SIU	2 SAG

10015	12 0	10076	LDA	0	SAG
	24 0	10073	MUI	0	FORH
10016	14 0	10063	ADD	0	COREB
	20 0	10034	STA	0	BMC
10017	56 3	10077	STJ	3	LAG
	12 0	10077	LDA	0	LAG
10020	24 0	10073	MUI	0	FORH
	14 0	10064	ADD	0	CORBT
10021	20 0	10035	STA	0	BTMC
	12 1	10300	LDA	1	10300
10022	05 0	00030	ALS	0	30
	14 0	10071	ADD	0	AINC
10023	20 0	10053	STA	0	LAA
	12 4	10300	LDA	4	10300
10024	05 0	00030	ALS	0	30
	14 0	10072	ADD	0	AAINC
10025	20 0	10054	STA	0	LAAA
	12 2	10400	LDA	2	10400
10026	05 0	00030	ALS	0	30
	14 0	10070	ADD	0	BINC
10027	20 0	10052	STA	0	LBA
	12 3	10400	LDA	3	10400
10030	20 0	10100	STA	0	LDD
	50 0	00000	ENI	0	0
10031	50 3	00000	ENI	3	0
	50 2	00000	ENI	2	0
10032	50 4	00000	ENI	4	0
	50 6	00000	ENI	6	0
10033	12 3	20002	AMC LDA	3	20002
	42 4	20002	SCM	4	20002
10034	42 2	30002	BMC SCM	2	30002
	53 6	10100	LIL	6	LDD
10035	64 6	30002	BTMC EQS	6	30002
	75 0	10052	SLJ	0	LBA
10036	42 0	10106	SCM	0	SVN
	75 4	07200	SLJ	4	FOURTH
10037	12 0	10076	LDA	0	SAG
	24 0	10073	MUI	0	FORH
10040	14 0	10065	ADD	0	BONP
	20 0	10041	STA	0	BNP
10041	12 2	00000	BNP LDA	2	0
	75 4	07214	SLJ	4	SECOND
10042	12 0	10074	LDA	0	BAG
	24 0	10105	MUI	0	TTHOU
10043	14 0	10066	ADD	0	ATWP
	20 0	10044	STA	0	ATNP
10044	12 4	00000	ATNP LDA	4	0
	75 4	07206	SLJ	4	THIRD
10045	12 0	10075	LDA	0	FAG
	24 0	10104	MUI	0	TWOH
10046	14 0	10067	ADD	0	AOWP
	20 0	10047	STA	0	AONP
10047	12 3	00000	AONP LDA	3	0
	75 4	07222	SLJ	4	FIRST
10050	57 2	10101	SIL	2	UNO
	57 3	10102	SIL	3	DOS

10051	57 4 10103		SIL	4 TRES
	75 0 07230		SLJ	0BEGINEQ
10052	54 2 00000	LBA	ISK	2 0
	75 0 10033		SLJ	0 AMC
10053	54 4 00000	LAA	ISK	4 0
	75 0 10033		SLJ	0 AMC
10054	54 3 00000	LAAA	ISK	3 0
	75 0 10033		SLJ	0 AMC
10055	53 1 10074		LIL	1 BAG
	52 4 10075		LIU	4 FAG
10056	52 2 10076		LIU	2 SAG
	52 3 10077		LIU	3 LAG
10057	75 0 10005		SLJ	0 BK
	50 0 00000		ENI	0 0
10060	53 2 10101	REL	LIL	2 UNO
	53 3 10102		LIL	3 DOS
10061	53 4 10103		LIL	4 TRES
	75 0 10052		SLJ	0 LBA
10062	12 3 20002	COREA	LDA	3 20002
	42 4 20002		SCM	4 20002
10063	42 2 30002	COREB	SCM	2 30002
	53 6 10100		LIL	6 LDD
10064	64 6 30002	CORBT	EQS	6 30002
	75 0 10052		SLJ	0 LBA
10065	12 2 30002	BONP	LDA	2 30002
	75 4 07214		SLJ	4 SECOND
10066	12 4 20002	ATWP	LDA	4 20002
	75 4 07206		SLJ	4 THIRD
10067	12 3 20002	AOWP	LDA	3 20002
	75 4 07222		SLJ	4 FIRST
10070	54 2 00000	BINC	ISK	2 0
	75 0 10033		SLJ	0 AMC
10071	54 4 00000	AINC	ISK	4 0
	75 0 10033		SLJ	0 AMC
10072	54 3 00000	AAINC	ISK	3 0
	75 0 10033		SLJ	0 AMC
10073	00 0 00000	FORH	OCT	400
	00 0 00400			
10074	00 0 00000	BAG	OCT	0
	00 0 00000			
10075	00 0 00000	FAG	OCT	0
	00 0 00000			
10076	00 0 00000	SAG	OCT	0
	00 0 00000			
10077	00 0 00000	LAG	OCT	0
	00 0 00000			
10100	00 0 00000	LDD	OCT	0
	00 0 00000			
10101	00 0 00000	UNG	OCT	0
	00 0 00000			
10102	00 0 00000	DOS	OCT	0
	00 0 00000			
10103	00 0 00000	TRES	OCT	0
	00 0 00000			
10104	00 0 00000	TWOH	OCT	200
	00 0 00200			

10105	00 0 00200	TTHOU OCT	20000000000
**	00 0 00000		
		ORG	7200
07200	75 0 00000	FOURTHSLJ	0
	04 0 00000	ENQ	0 0
07201	03 0 00015	LRS	0 15
	50 5 00000	ENI	5 00000
07202	07 0 00001	FOURTHLLS	0 1
	20 5 07101	STA	5 7101
07203	10 0 00000	ENA	0 0
	51 5 00001	INI	5 1
07204	54 5 99931	THRU4 ISK	5 31
	75 0 07202	SLJ	0 FOURTHL
07205	75 0 07200	SLJ	0 FOURTH
	50 0 00000	ENI	0 0
07206	75 0 00000	THIRD SLJ	0
	04 0 00000	ENQ	0 0
07207	03 0 00015	LRS	0 15
	50 5 00030	ENI	5 30
07210	07 0 00001	THIRDL LLS	0 1
	20 5 07100	STA	5 7100
07211	51 5 77774	INI	5 77774
	10 0 00000	ENA	0 0
07212	54 4 77774	THRU3 ISK	5 77774
	75 0 07210	SLJ	0 THIRDL
07213	75 0 07206	SLJ	0 THIRD
	50 0 00000	ENI	0 0
07214	75 0 00000	SECOND SLJ	0
	04 0 00000	ENQ	0 0
07215	03 0 00015	LRS	0 15
	50 5 00000	ENI	5 00000
07216	07 0 00001	SECONDL LLS	0 1
	20 5 07001	STA	5 7001
07217	10 0 00000	ENA	0 0
	51 5 00001	INI	5 1
07220	54 5 00031	THRU2 ISK	5 31
	75 0 07216	SLJ	0 SECONDL
07221	75 0 07214	SLJ	0 SECOND
	50 0 00000	ENI	0 0
07222	75 0 00000	FIRST SLJ	0
	04 0 00000	ENQ	0 0
07223	03 0 00015	LRS	0 15
	50 5 00030	ENI	5 30
07224	07 0 00001	FIRSTL LLS	0 1
	20 5 07000	STA	5 7000
07225	51 5 77774	INI	5 77774
	10 0 00000	ENA	0 0
07226	54 5 77774	THRU1 ISK	5 77774
	75 0 07224	SLJ	0 FIRSTL
07227	75 0 07222	SLJ	0 FIRST
	50 0 00000	ENI	0 0
07230	10 0 00032	BEGINEQ ENA	0 32
	20 0 07247	STA	0 N
07231	11 0 77776	INA	0 77776
	60 0 07233	SAU	0 BEGIN

07232	60 0 07243	SAU	0	CHECKEQ
	50 4 00000	ENI	4	0
07233	50 1 00000	BEGIN ENI	1	
	50 0 00000	ENI	0	0
07234	57 1 07251	BIGLOOP SIL	1	TEMP
	50 3 00000	ENI	3	00000
07235	50 2 00000	ENI	2	00000
	53 4 07251	LIL	4	TEMP
07236	12 2 07000	SMALLER LDA	2	7000
	15 4 07000	SUB	4	7000
07237	22 1 07240	AJP	1	BSECTION
	51 3 00001	INI	3	00001
07240	12 2 07100	BSECTION LDA	2	7100
	15 4 07100	SUB	4	7100
07241	22 0 07242	AJP	0	ADV
	51 3 77776	INI	3	77776
07242	51 2 00001	ADV INI	2	00001
	50 0 00000	ENI	0	0
07243	54 4 00000	CHECKEQ ISK	4	
	75 0 07236	SLJ	0	SMALLER
07244	55 3 10060	IJP	3	REL
	51 1 77776	INI	1	77776
07245	55 1 07246	IJP	1	RESET
	76 0 064000	SLS	0	64000
07246	51 1 00001	RESET INI	1	1
	75 0 07234	SLJ	0	BIGLOOP
07247	00 0 00000	N	DEC	
	00 0 00000			
07250	00 0 00000	NMINUS1	DEC	
	00 0 00000			
07251	00 0 00000	TEMP	DEC	
	00 0 00000			
07252	00 0 00000	ONE	DEC	
	00 0 00000			
		ORG	10300	
10300	00 0 00000	OCT	16	
	00 0 00016			
10301	00 0 00000	OCT	16	
	00 0 00016			
10302	00 0 00000	OCT	34	
	00 0 00034			
10303	00 0 00000	OCT	54	
	00 0 00054			
10304	00 0 00000	OCT	54	
	00 0 00054			
10305	00 0 00000	OCT	54	
	00 0 00054			
10306	00 0 00000	OCT	26	
	00 0 00026			
10307	00 0 00000	OCT	130	
	00 0 00130			
10310	00 0 00000	OCT	26	
	00 0 00026			
10311	00 0 00000	OCT	16	
	00 0 00016			

10312	00 0 00000	OCT	142
	00 0 00142		
10313	00 0 00000	OCT	16
	00 0 00016		
10314	00 0 00000	OCT	34
	00 0 00034		
10315	00 0 00000	OCT	106
	00 0 00106		
10316	00 0 00000	OCT	106
	00 0 00106		
10317	00 0 00000	OCT	54
	00 0 00054		
10320	00 0 00000	OCT	26
	00 0 00026		
10321	00 0 00000	OCT	43
	00 0 00043		
		ORG	10400
10400	00 0 00000	OCT	106
	00 0 00106		
10401	00 0 00000	OCT	106
	00 0 00106		
10402	00 0 00000	OCT	214
	00 0 00214		
10403	00 0 00000	OCT	160
	00 0 00160		
10404	00 0 00000	OCT	160
	00 0 00160		
10405	00 0 00000	OCT	160
	00 0 00160		
10406	00 0 00000	OCT	70
	00 0 00070		
10407	00 0 00000	OCT	340
	00 0 00340		
10410	00 0 00000	OCT	70
	00 0 00070		
10411	00 0 00000	OCT	106
	00 0 00106		
10412	00 0 00000	OCT	304
	00 0 00304		
10413	00 0 00000	OCT	106
	00 0 00106		
10414	00 0 00000	OCT	214
	00 0 00214		
10415	00 0 00000	OCT	70
	00 0 00070		
10416	00 0 00000	OCT	70
	00 0 00070		
10417	00 0 00000	OCT	160
	00 0 00160		
10420	00 0 00000	OCT	70
	00 0 00070		
10421	00 0 00000	OCT	10
	00 0 00010		
		ORG	15500
15500	00 0 00000	OCT	0
	00 0 00000		

15501	00 0 00000	OCT	20
	00 0 00020		
15502	00 0 00000	OCT	110
	00 0 00110		
15503	00 0 00000	OCT	1
	00 0 00001		
15504	00 0 00000	OCT	11
	00 0 00011		
15505	00 0 00000	OCT	21
	00 0 00021		
15506	00 0 00000	OCT	101
	00 0 00101		
15507	00 0 00000	OCT	111
	00 0 00111		
15510	00 0 00000	OCT	121
	00 0 00121		
15511	00 0 00000	OCT	2
	00 0 00002		
15512	00 0 00000	OCT	12
	00 0 00012		
15513	00 0 00000	OCT	22
	00 0 00022		
15514	00 0 00000,	OCT	102
	00 0 00102		
15515	00 0 00000	OCT	112
	00 0 00112		
15516	00 0 00000	OCT	122
	00 0 00122		
15517	00 0 00000	OCT	13
	00 0 00013		
15520	00 0 00000	OCT	103
	00 0 00103		
15521	00 0 00000	OCT	123
	00 0 00123		
		ORG	16000
16000	00 0 00000	OCT	123
	00 0 00123		
16001	00 0 00000	OCT	123
	00 0 00123		
16002	00 0 00000	OCT	12
	00 0 00012		
16003	00 0 00000	OCT	12
	00 0 00012		
16004	00 0 00000	OCT	12
	00 0 00012		
16005	00 0 00000	OCT	12
	00 0 00012		
16006	00 0 00000	OCT	12
	00 0 00012		
16007	00 0 00000	OCT	12
	00 0 00012		
16010	00 0 00000	OCT	12
	00 0 00012		
16011	00 0 00000	OCT	12
	00 0 00012		

16012	00 0 00000	OCT	112
	00 0 00112		
16013	00 0 00000	OCT	112
	00 0 00112		
16014	00 0 00000	OCT	112
	00 0 00112		
16015	00 0 00000	OCT	112
	00 0 00112		
16016	00 0 00000	OCT	122
	00 0 00122		
16017	00 0 00000	OCT	122
	00 0 00122		
16020	00 0 00000	OCT	122
	00 0 00122		
16021	00 0 00000	OCT	122
	00 0 00122		
16022	00 0 00000	OCT	1
	00 0 00001		
16023	00 0 00000	OCT	1
	00 0 00001		
16024	00 0 00000	OCT	1
	00 0 00001		
16025	00 0 00000 ,	OCT	1
	00 0 00001		
16026	00 0 00000	OCT	11
	00 0 00011		
16027	00 0 00000	OCT	11
	00 0 00011		
16030	00 0 00000	OCT	11
	00 0 00011		
16031	00 0 00000	OCT	11
	00 0 00011		
16032	00 0 00000	OCT	13
	00 0 00013		
16033	00 0 00000	OCT	13
	00 0 00013		
16034	00 0 00000	OCT	13
	00 0 00013		
16035	00 0 00000	OCT	13
	00 0 00013		
16036	00 0 00000	OCT	21
	00 0 00021		
16037	00 0 00000	OCT	21
	00 0 00021		
16040	00 0 00000	OCT	21
	00 0 00021		
16041	00 0 00000	OCT	21
	00 0 00021		
16042	00 0 00000	OCT	101
	00 0 00101		
16043	00 0 00000	OCT	101
	00 0 00101		
16044	00 0 00000	OCT	111
	00 0 00111		

16045	00 0 00000	OCT	111
	00 0 00111		
16046	00 0 00000	OCT	111
	00 0 00111		
16047	00 0 00000	OCT	111
	00 0 00111		
16050	00 0 00000	OCT	111
	00 0 00111		
16051	00 0 00000	OCT	111
	00 0 00111		
16052	00 0 00000	OCT	111
	00 0 00111		
16053	00 0 00000	OCT	111
	00 0 00111		
16054	00 0 00000	OCT	103
	00 0 00103		
16055	00 0 00000	OCT	103
	00 0 00103		
16056	00 0 00000	OCT	121
	00 0 00121		
16057	00 0 00000	OCT	121
	00 0 00121		
16060	00 0 00000,	OCT	0
	00 0 00000		
16061	00 0 00000	OCT	0
	00 0 00000		
16062	00 0 00000	OCT	2
	00 0 00002		
16063	00 0 00000	OCT	2
	00 0 00002		
16064	00 0 00000	OCT	110
	00 0 00110		
16065	00 0 00000	OCT	110
	00 0 00110		
16066	00 0 00000	OCT	110
	00 0 00110		
16067	00 0 00000	OCT	110
	00 0 00110		
16070	00 0 00000	OCT	20
	00 0 00020		
16071	00 0 00000	OCT	20
	00 0 00020		
16072	00 0 00000	OCT	102
	00 0 00102		
16073	00 0 00000	OCT	102
	00 0 00102		
16074	00 0 00000	OCT	102
	00 0 00102		
16075	00 0 00000	OCT	102
	00 0 00102		
16076	00 0 00000	OCT	22
	00 0 00022		
16077	00 0 00000	OCT	22
	00 0 00022		

15000	00 0 00000	ORG 15000
	00 0 00000	OCT 0
15001	00 0 00000	OCT 16007
	00 0 16007	
150002	00 0 00000	OCT 10000
	00 0 10000	
15003	00 0 00000	OCT 1
	00 0 00001	
15004	00 0 00000	OCT 14002
	00 0 14002	
15005	00 0 00000	OCT 12004
	00 0 12004	
15006	00 0 00000	OCT 04003
	00 0 04003	
15007	00 0 00000	OCT 2005
	00 0 02005	
15010	00 0 00000	OCT 16006
	00 0 16006	
15011	00 0 00000	OCT 6007
	00 0 06007	
15012	00 0 00000	OCT 4000
	00 0 04000	
15013	00 0 00000	OCT 2
	00 0 00002	
15014	00 0 00000	OCT 16005
	00 0 16005	
15015	00 0 00000	OCT 12007
	00 0 12007	
15016	00 0 00000	OCT 2000
	00 0 02000	
15017	00 0 00000	OCT 4
	00 0 00004	
15020	00 0 00000	OCT 16003
	00 0 16003	
15021	00 0 00000	OCT 14007
	00 0 14007	
15022	00 0 00000	OCT 14000
	00 0 14000	
15023	00 0 00000	OCT 3
	00 0 00003	
15024	00 0 00000	OCT 16004
	00 0 16004	
15025	00 0 00000	OCT 2007
	00 0 02007	
15026	00 0 00000	OCT 12000
	00 0 12000	
15027	00 0 00000	OCT 5
	00 0 00005	
15030	00 0 00000	OCT 16002
	00 0 16002	
15031	00 0 00000	OCT 4007
	00 0 04007	
15032	00 0 00000	OCT 10004
	00 0 10004	

15033	00 0 00000	OCT	2001
	00 0 02001		
15034	00 0 00000	OCT	14006
	00 0 14006		
15035	00 0 00000	OCT	6003
	00 0 06003		
15036	00 0 00000	OCT	10002
	00 0 10002		
15037	00 0 00000	OCT	4001
	00 0 04001		
15040	00 0 00000	OCT	12006
	00 0 12006		
15041	00 0 00000	OCT	6005
	00 0 06005		
15042	00 0 00000	OCT	10001
	00 0 10001		
15043	00 0 00000	OCT	6006
	00 0 06006		
15044	00 0 00000	OCT	6000
	00 0 06000		
15045	00 0 00000	OCT	4004
	00 0 04004		
15046	00 0 00000,	OCT	2002
	00 0 02002		
15047	00 0 00000	OCT	6
	00 0 00006		
15050	00 0 00000	OCT	16001
	00 0 16001		
15051	00 0 00000	OCT	14005
	00 0 14005		
15052	00 0 00000	OCT	12003
	00 0 12003		
15053	00 0 00000	OCT	10007
	00 0 10007		
15054	00 0 00000	OCT	4002
	00 0 04002		
15055	00 0 00000	OCT	12005
	00 0 12005		
15056	00 0 00000	OCT	2004
	00 0 02004		
15057	00 0 00000	OCT	14003
	00 0 14003		
15060	00 0 00000	OCT	16000
	00 0 16000		
15061	00 0 00000	OCT	7
	00 0 00007		
15062	00 0 00000	OCT	14004
	00 0 14004		
15063	00 0 00000	OCT	2003
	00 0 02003		
15064	00 0 00000	OCT	14001
	00 0 14001		
15065	00 0 00000	OCT	10003
	00 0 10003		

15066	00 0 00000	OCT	6004
	00 0 06004		
15067	00 0 00000	OCT	2006
	00 0 02006		
15070	00 0 00000	OCT	12002
	00 0 12002		
15071	00 0 00000	OCT	4005
	00 0 04005		
15072	00 0 00000	OCT	12001
	00 0 12001		
15073	00 0 00000	OCT	10005
	00 0 10005		
15074	00 0 00000	OCT	6002
	00 0 06002		
15075	00 0 00000	OCT	4006
	00 0 04006		
15076	00 0 00000	OCT	10006
	00 0 10006		
15077	00 0 00000	OCT	6001
	00 0 06001		

END

**

10106	00 0 00000	SVN	OCT	17777
	00 0 17777			

APPENDIX VI

				ORG	7400
07400	50	1	00000	ERAS	ENI 1 0
	50	2	00000		ENI 2 0
07401	50	3	00000		ENI 3 0
	50	4	00000		ENI 4 0
07402	50	5	00000		ENI 5 0
	50	6	00000		ENI 6 0
07403	10	0	00000		ENA 0 0
	20	1	600000		STA 1 60000
07404	54	1	00210		ISK 1 2110
	75	0	07403		SLJ 0 /-1
07405	50	0	00000		ENI 0 0
	53	1	50107		LJL 1 RETNA
07406	54	1	00227		ISK 1 227
	75	0	07410		SLJ 0 BUS
07407	76	0	06655		SLS 0 6655
	50	0	00000		ENI 0 0
07410	57	1	50107	BUS	SIL 1 RETNA
	50	1	00000		ENI 1 0
07411	75	4	51000		SLJ 4 INPU
	50	0	00000		ENI 0 0
07412	75	5	51032		SLJ 5 VERT
	12	0	50105,		LDA 0 DUM7
07413	20	0	07426		STA 0 SERE- 1
	50	0	00000		ENI 0 0
07414	12	0	60200		LDA 0 60200
	22	0	07460		AJP 0 TAT
07415	10	0	00210		ENA 0 210
	20	0	07733		STA 0 TEM1
07416	60	0	07450		SAU 0 JJ
	10	0	00007		ENA 0 7
07417	20	0	07734		STA 0 TEM2
	60	0	07723		SAU 0 GOF
07420	50	0	00000	UU	ENI 0 0
	50	2	00001		ENI 2 1
07421	04	0	00001	VV	ENQ 0 1
	50	0	00000		ENI 0 0
07422	06	2	00000		QLS 2 0
	44	1	60000		LDL 1 60000
07423	01	2	00000		ARS 2 0
	20	0	07457		STA 0 PERE
07424	04	0	00001		ENQ 0 1
	06	3	00000		QLS 3 0
07425	44	1	60000		LDL 1 60000
	01	3	00000		ARS 3 0
07426	42	0	07457		SCM 0 PERE
	22	0	07454		AJP 0 FRERE
07427	51	2	00001	SERE	INI 2 1
	50	0	00000		ENI 0 0
07430	54	3	00020		ISK 3 20
	75	0	07421		SLJ 0 VV
07431	72	0	07420		RAO 0 UU
	50	0	00000		ENI 0 0
07432	54	4	00006		ISK 4 6
	75	0	07420		SLJ 0 UU

07433	12 4 07600	WW	LDA	4	MERE
	50 0 00000		ENI	0	0
07434	05 4 00000		ALS	4	0
	05 4 00000		ALS	4	0
07435	05 4 00000		ALS	4	0
	05 4 00000		ALS	4	0
07436	05 4 00000		ALS	4	0
	05 4 00000		ALS	4	0
07437	20 4 07600		STA	4	MERE
	50 0 00000		ENI	0	0
07440	54 4 00006		ISK	4	6
	75 0 07433		SLJ	0	WW
07441	10 0 00000		ENA	0	0
	50 0 00000		ENI	0	0
07442	14 4 07600	XX	ADD	4	MERE
	50 0 00000		ENI	0	0
07443	54 4 00006		ISK	4	6
	75 0 07442		SLJ	0	XX
07444	20 1 61000		STA	1	61000
	50 0 00000		ENI	0	0.
07445	12 0 07455		LDA	0	YY
	20 0 07420		STA	0	UU
07446	10 0 0000Q	ZZ	ENA	0	0
	20 4 07600		STA	4	MERE
07447	54 4 00006		ISK	4	6
	75 0 07446		SLJ	0	ZZ
07450	54 1 00000	JJ	ISK	1	0
	75 0 07420		SLJ	0	UU
07451	75 1 51034		SLJ	1	TERN
	12 0 07734		LDA	0	TEM2
07452	20 0 63100		STA	0	63100
	50 5 00000		ENI	5	0
07453	75 0 07700		SLJ	0	YEA
	50 0 00000		ENI	0	0
07454	72 4 07600	FRERE	RAO	4	MERE
	75 0 07427		SLJ	0	SERE
07455	50 0 00000	YY	ENI	0	0
	50 2 00001		ENI	2	1
07456	00 0 00000	THSV	OCT		377777
	00 3 77777				
07457	00 0 00000	PERE	OCT		0
	00 0 00000				
07460	10 0 00104	TAT	ENA	0	104
	20 0 07733		STA	0	TEM1
07461	60 0 07450		SAU	0	JJ
	10 0 00003		ENA	0	3
07462	20 0 07734		STA	0	TEM2
	60 0 07723		SAU	0	GOF
07463	75 0 07420		SLJ	0	UU
	50 0 00000		FNI	0	0
			ORG		7600
07600	00 0 00000	MERE	OCT		0
	00 0 00000				

				ORG	7700
07700	50 1 00000	YEA		ENI	1 0
	50 4 00000			ENI	4 0
07701	50 3 00000			ENI	3 0
	50 6 00000			ENI	6 0
07702	53 2 07733			LIL	2 TEM1
	56 2 07725			SIU	2 ELSE
07703	12 0 07733			LDA	0 TEM1
	11 0 77776			INA	0 -1
07704	60 0 07726			SAU	0 PATE
	50 0 00000			ENI	0 0
07705	12 1 61000	RGE		LDA	1 61000
	22 0 07726			AJP	0 PATE
07706	20 5 63000			STA	5 63000
	57 3 07730			SIL	3 ERE
07707	64 2 61000	BALE		EQS	2 61000
	75 0 07725			SLJ	0 ELSE
07710	10 0 00000			ENA	0 0
	20 2 61000			STA	2 61000
07711	12 0 07730			LDA	0 ERE
	24 0 07731			MUI	0 RTY
07712	14 0 07732			ADD	0 TAP
	20 0 07714.			STA	0 TUB
07713	16 0 07456			LDQ	0 THSV
	50 0 00000			ENI	0 0
07714	44 2 60000	TUB		LDL	2 60000
	20 4 62000			STA	4 62000
07715	12 1 61000			LDA	1 61000
	50 0 00000			ENI	0 0
07716	54 4 00020			ISK	4 20
	75 0 07707			SLJ	0 BALE
07717	51 3 00001			INI	3 1
	51 5 00001			INI	5 1
07720	57 2 50104			SIL	2 DOL
	10 0 00000			ENA	0 0
07721	65 2 61000			THS	2 61000
	75 0 07725			SLJ	0 ELSE
07722	53 2 50104			LIL	2 DOL
	50 0 00000			ENI	0 0
07723	54 6 00000	GOF		ISK	6 0
	75 0 07705			SLJ	0 RGE
07724	76 0 04444			SLS	0 4444
	50 0 00000			ENI	0 0
07725	50 2 00000	ELSE		ENI	2 0
	55 4 07735			IJP	4 SIGNAL
07726	54 1 00000	PATE		ISK	1 0
	75 0 07705			SLJ	0 RGE
07727	75 1 07742			SLJ	1 FOP2
	75 0 07737			SLJ	0 FOP1
07730	00 0 00000	ERE		OCT	0
	00 0 00000				
07731	00 0 00000	RTY		OCT	100
	00 0 00100				
07732	44 2 60000	TAP		LDL	2 60000
	20 4 62000			STA	4 62000

07733	00 0 00000	TEM1	OCT	0
	00 0 00000			
07734	00 0 00000	TEM2	OCT	0
	00 0 00000			
07735	04 0 77777	SIGNAL	ENQ	0 77777
	76 0 02040		SLS	0 2040
07736	51 1 00001	ELSE1	INI	1 1
	75 0 07705		SLJ	0 RGE
07737	12 1 62000	FOP1	LDA	1 62000
	20 1 10000		STA	1 A
07740	54 1 00777		ISK	1 777
	75 0 07737		SLJ	0 FOP1
07741	76 0 00111		SLS	0 111
	50 0 00000		ENI	0 0
		A	EQU	10000
07742	12 1 62000	FOP2	LDA	1 62000
	20 1 11000		STA	1 B
07743	54 1 00777		ISK	1 777
	75 0 07742		SLJ	0 FOP2
07744	75 0 50000		SLJ	0 50000
	50 0 00000		ENI	0 0
			ORG	50000
50000	50 3 00000,		ENI	3 0
	50 4 00000		ENI	4 0
50001	50 5 00000		ENI	5 0
	60 6 00000		ENI	6 0
50002	53 1 63100		LIL	1 63100
	53 2 63101		LIL	2 63101
50003	56 1 50061		SIU	1 INFC
	56 1 50060		SIU	1 INFB
50004	56 2 50057		SIU	2 INFA
	53 4 63101		LIL	4 63101
50005	50 1 00000		ENI	1 0
	50 2 00000		ENI	2 0
50006	50 3 00000		ENI	3 0
	50 0 00000		ENI	0 0
50007	12 1 63000	JMP2	LDA	1 63000
	14 2 63000		ADD	2 63000
50010	15 3 63010		SUB	3 63010
	50 0 00000		ENI	0 0
50011	64 4 63010	JMP1	EQS	4 63010
	75 0 50063		SLJ	0 SWHE
50012	56 1 50073		SIU	1 TE3
	57 2 50100		SIL	2 TE4
50013	56 3 50101		SIU	3 TE5
	56 4 50102		SIU	4 TE6
50014	50 4 00020		ENI	4 20
	50 1 00000		ENI	1 0
50015	50 2 00000		ENI	2 0
	50 3 00000		ENI	3 0
50016	12 0 50073		LDA	0 TE3
	24 0 07731		MUI	0 RTY
50017	14 0 50064		ADD	0 DUM1
	20 0 50103		STA	0 DEL

50020	12 0 50100		LDA	0 TE4
	24 0 07731		MUI	0 RTY
50021	70 0 50103		RAD	0 DEL
	20 0 50030		STA	0 INF1
50022	12 0 50101		LDA	0 TE5
	24 0 07731		MUI	0 RTY
50023	14 0 50065		ADD	0 DUM2
	20 0 50031		STA	0 INF2
50024	12 0 50102		LDA	0 TE6
	24 0 07731		MUI	0 RTY
50025	14 0 50065		ADD	0 DUM2
	20 0 50031		STA	0 INF2
50026	12 0 50102		LDA	0 TE6
	24 0 07731		MUI	0 RTY
50027	14 0 50066		ADD	0 DUM3
	20 0 50032		STA	0 INF3
50030	12 1 10000	INF1	LDA	1 A
	42 2 10000		SCM	2 A
50031	42 3 11000	INF2	SCM	3 B
	50 4 00020		ENI	4 20
50032	64 4 11000	INF3	EQS	4 B
	75 0 50052		SLJ	0 INSKIP
50033	42 0 50067		SCM	0 TRENSE
	75 4 00017		SLJ	4 FOURTH
50034	12 0 50100		LDA	0 TE4
	05 0 00036		ALS	0 36
50035	14 0 50070		ADD	0 DUM4
	20 0 50036		STA	0 INF4
50036	12 2 10000	INF4	LDA	2 A
	75 4 00016		SLJ	4 THIRD
50037	12 0 50101		LDA	0 TE5
	05 0 00006		ALS	0 6
50040	14 0 50071		ADD	0 DUM5
	20 0 50041		STA	0 INF5
50041	12 3 11000	INF5	LDA	3 B
	75 4 00015		SLJ	4 SECOND
50042	12 0 50073		LDA	0 TE3
	05 0 00006		ALS	0 6
50043	14 0 50072		ADD	0 DUM6
	20 0 50044		STA	0 INF6
50044	12 1 10000	INF6	LDA	1 A
	75 4 00014		SLJ	4 FIRST
50045	57 1 50074		SIL	1 FAG
	57 2 50075		SIL	2 SAG
50046	57 3 50076		SIL	3 LAG
	57 4 50077		SIL	4 GAG
50047	75 0 00013		SLJ	0 BEGINEQ
	50 0 00000		ENI	0 0
50050	53 1 50074	REL	LIL	1 FAG
	53 2 50075		LIL	2 SAG
50051	53 3 50076		LIL	3 LAG
	53 4 50077		LIL	4 GAG
50052	54 3 00020	INSKIP	ISK	3 20
	75 0 50030		SLJ	0 INF1

50053	54 2 00020		ISK	2 20
	75 0 50030		SLJ	0 INF1
50054	54 1 00020		ISK	1 20
	75 0 50030		SLJ	0 INF1
50055	52 1 50073		LIU	1 TE3
	53 2 50100		LIL	2 TE4
50056	52 3 50101		LIU	3 TE5
	52 4 50102		LIU	4 TE6
50057	54 3 00000	INFA	ISK	3 0
	75 0 50007		SLJ	0 JMP2
50060	54 2 00000	INFB	ISK	2 0
	75 0 50007		SLJ	0 JMP2
50061	54 1 00000	INFC	ISK	1 0
	75 0 50007		SLJ	0 JMP2
50062	76 0 01111		SLS	0 1111
	50 0 00000		ENI	0 0
50063	53 4 63101	SWHE	LIL	4 63101
	75 0 50057		SLJ	0 INFA
50064	12 1 10000	DUM1	LDA	1 A
	42 2 10000		SCM	2 A
50065	42 3 11000	DUM2	SCM	3 B
	50 4 00020		ENI	4 20
50066	64 4 11000	DUM3	EQS	4 B
	75 0 50052		SLJ	0 INSKIP
50067	00 0 00000	TRENSE	OCT	377777
	00 3 77777			
50070	12 2 100000	DUM4	LDA	2 A
	75 4 00012		SLJ	4 THIRD
50071	12 3 11000	DUM5	LDA	3 B
	75 4 00011		SLJ	4 SECOND
50072	12 1 10000	DUM6	LDA	1 A
	75 4 00010		SLJ	4 FIRST
50073	00 0 00000	TE3	OCT	0
	00 0 00000			
50074	00 0 00000	FAG	OCT	0
	00 0 00000			
50075	00 0 00000	SAG	OCT	0
	00 0 00000			
50076	00 0 00000	LAG	OCT	0
	00 0 00000			
50077	00 0 00000	GAG	OCT	0
	00 0 00000			
50100	00 0 00000	TE4	OCT	0
	00 0 00000			
50101	00 0 00000	TE5	OCT	0
	00 0 00000			
50102	00 0 00000	TE6	OCT	0
	00 0 00000			
50103	00 0 00000	DEL	OCT	0
	00 0 00000			
		B	EQU	11000
50104	00 0 00000	DOL	OCT	0
	00 0 00000			
50105	42 0 07457	DUM7	SCM	0 PERE
	22 0 07454		AJP	0 FRERE

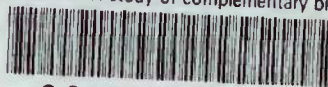
50106	42 0 07457	DUM10	SCM	0 PERE
	22 1 07454		AJP	1 FRERE
50107	00 0 00000	RETNA	OCT	0
	00 0 00000			
50110	00 0 00000	ANTER	OCT	0
	00 0 00000			
			ORG	51000
51000	75 0 00000	INPU	SLJ	0 0
	50 0 00000		ENI	0 0
51001	57 6 51020		SIL	6 CONT
	50 6 00004		ENI	6 4
51002	50 0 00000		ENI	0 0
	10 0 00000		ENA	0 0
51003	74 0 52031		EXF	0 52031
	74 7 52000		EXF	7 52000
51004	20 0 57777		STA	0 57777
	10 0 60001		ENA	0 60001
51005	61 0 00005		SAL	0 5
	74 5 57777		EXF	5 57777
51006	12 0 57777		LDA	0 57777
	22 0 51006		AJP	0 /
51007	14 0 51014		ADD	0 CONSI
	20 0 00005,		STA	0 5
51010	50 0 00000		ENI	0 0
	74 7 52000		EXF	7 52000
51011	74 7 52003		EXF	7 52003
	75 0 51015		SLJ	0 ERR
51012	74 7 52005		EXF	7 52005
	75 0 51015		SLJ	0 ERR
51013	53 6 51020		LIL	6 CONT
	75 0 51000		SLJ	0 INPU
51014	00 0 00001	CONSI	ZRO	0 1
	00 0 00000		ZRO	0 0
51015	55 6 51016	ERR	IJP	6 /+1
	75 0 51036		SLJ	0 EREX
51016	74 0 52006		EXF	0 52006
	74 7 52000		EXF	7 52000
51017	75 0 51002		SLJ	0 INPU+2
	50 0 00000		ENI	0 0
51020	00 0 00000	CONT	OCT	0
	00 0 00000			
51021	50 0 00000		ENI	0 0
	50 0 00000			
51022	50 0 00000	OUTPUT	ENI	0 0
	74 0 52000		EXF	0 52000
51023	57 1 51030		SIL	1 /+5
	55 1 51025		IJP	1 /+2
51024	75 0 51031		SLJ	0 RTN2
	50 0 00000		ENI	0 0
51025	10 0 60000		ENA	0 60000
	61 0 00005		SAL	0 5
51026	74 0 52031		EXF	0 52031
	74 7 52000		EXF	7 52000
51027	74 5 57777		EXF	5 57777
	74 7 52000		EXF	7 52000

51030	55	1	51027		IJP	1	/-1
	50	0	00000		ENI	0	0
51031	50	0	00000	RTN2	ENI	0	0
	76	0	02134		SLS	0	2134
51032	75	0	00000	VERT	SLJ	0	0
	50	0	00000		ENI	0	0
51033	12	0	50106		LDA	0	DUM10
	75	0	51032		SLJ	0	VERT
51034	50	5	00010	TERN	ENI	5	10
	12	0	07734		LDA	0	TEM2
51035	20	0	63101		STA	0	63101
	75	0	07700		SLJ	0	YEA
51036	12	0	50107	EREX	LDA	0	RETNA
	53	5	50110		LIL	5	ANTER
51037	20	5	65000		STA	5	HOPET
	51	5	00001		INI	5	1
51040	57	5	50110		SIL	5	ANTER
	50	5	00000		ENI	5	0
51041	75	0	07400		SLJ	0	ERAS
	50	0	00000		ENI	0	0.
				HOPET	EQU		65000
					END		

2

thesJ339

A theoretical study of complementary bin



3 2768 002 10045 5

DUDLEY KNOX LIBRARY